Universidad Carlos III de Madrid

Escuela Politécnica Superior

Grado en Ingeniería en Tecnologías Industriales

Trabajo de Fin de Grado

Agentes robóticos y entretenimiento

Autor: Jesús Naranjo Bolaños

Tutor: Dr. D. Daniel Borrajo Millán

Año de edición: 2015

RESUMEN

En el presente proyecto se ha desarrollado una aplicación que permite al robot humanoide *NAO* interpretar melodías mediante el uso de un teclado electrónico. Con esto se pretende ampliar las ya numerosas capacidades del robot NAO así como estudiar la viabilidad de su utilización para la docencia en el campo de la música.

Para alcanzar este objetivo, se han utilizado los lenguajes *Python* y *XML* así como el simulador *Choregraphe* proporcionado por *Aldebaran Robotics* y el software libre *MuseScore*.

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

1 - INTRODUCCIÓN

En este capítulo introductorio, tras una descripción general del proyecto "Agentes robóticos y entretenimiento", se exponen las motivaciones que han llevado a su realización para, posteriormente, presentar la estructura utilizada para la elaboración del presente documento, correspondiente a la memoria del proyecto.

1.1 – DESCRIPCIÓN GENERAL DEL PROYECTO

Por medio de la realización de este proyecto se pretende conseguir que un robot humanoide *NAO* sea capaz de interpretar melodías mediante el uso de un teclado electrónico al tiempo que proporciona, al usuario de la aplicación, información relativa a las notas interpretadas con el objetivo de poder realizar una labor educativa.

1.2 - MOTIVACIÓN

A lo largo de los últimos años, se han producido grandes avances en el campo de la robótica, tanto en el hardware del que se componen estos elementos robóticos como en el software necesario para gestionarlos.

Gracias a estos avances, se ha producido un considerable incremento en el uso de los robots con fines educativos o sencillamente con el objetivo de crear entretenimiento para niños y adultos.

Para la realización de estas nuevas labores, los conocidos como robots sociales han resultado ser los más adecuados. Estos robots tienen una muy evolucionada capacidad de interacción con el entorno, lo que les permite comunicarse con su usuario de manera sencilla y agradable.

El robot humanoide *NAO* creado por la empresa *Aldebaran*, gracias al elevado número de actuadores y captadores que posee y a sus funciones de reconocimiento de voz, resulta muy adecuado para actuar como robot social. Además de sus capacidades, el diseño estético del *NAO* logra que se asemeje a un ser humano de dimensiones reducidas lo cual hace que su uso con fines de ocio resulte más atractivo para sus potenciales usuarios.

Esta ventaja estética del robot *NAO* es la que ha motivado, en el caso de este proyecto, su utilización en el campo del ocio y, por otra parte, han sido preferencias personales los que han llevado a la determinación de generar entretenimiento musical pudiendo, además, realizar labores educativas.

1.3 – ESTRUCTURA DEL DOCUMENTO

El documento se ha dividido en capítulos, secciones y subsecciones. A continuación, se describe brevemente cuáles son estos capítulos y cuál es la temática tratada en cada uno de ellos.

- **Capítulo 1 - Introducción.** Se exponen las motivaciones que han llevado al desarrollo del proyecto y se presenta la estructura de la presente "Memoria de Proyecto".

- **Capítulo 2 - Estado de la cuestión.** Se analizan los antecedentes que han llevado a la situación actual en el campo de la robótica así como las características de los elementos más relevantes para el desarrollo de la aplicación objetivo. Por último, se muestran otros proyectos que han servido de inspiración durante el establecimiento de las características principales pretendidas para la aplicación final.

- **Capítulo 3 - Objetivos.** Se exponen los objetivos principales que se pretende alcanzar mediante la realización de este proyecto.

- **Capítulo 4 - Características de los complementos.** Dadas las cualidades altamente personalizables de los complementos, se ha considerado conveniente la elaboración de un capítulo que muestre de manera específica cuáles son las características escogidas para la realización específica de este proyecto.

- **Capítulo 5 - Diseño y desarrollo de la aplicación.** Se muestran la arquitectura de la aplicación final y su diagrama de flujo para, posteriormente, exponer el proceso de desarrollo y los métodos empleados para su elaboración. Finalmente, se analizan las características básicas que deben poseer las partituras para que puedan ser leídas por la aplicación e interpretadas por el robot.

- **Capítulo 6 - Manual de usuario.** Se trata de un manual de instrucciones cuyo objetivo es guiar al usuario de la aplicación para su utilización de la forma más adecuada consiguiendo, de este modo, obtener un correcto funcionamiento de la misma.

- **Capítulo 7 - Manual de referencia.** Se trata de una guía cuyo objetivo es orientar a un futuro desarrollador que deseara ampliar la aplicación para que éste pueda comprender sencilla y rápidamente el funcionamiento del programa.

- **Capítulo 8 - Gestión del trabajo.** Se analizan, de manera detallada, la planificación del proyecto y el presupuesto requerido para la realización del mismo.

- **Capítulo 9 - Resultados.** Se comprueba el grado de éxito con que se han alcanzado los diversos objetivos que fueron planteados al comienzo del proyecto.

- **Capítulo 10 - Conclusiones.** Se exponen las conclusiones obtenidas a partir del análisis de los resultados obtenidos.

- **Capítulo 11 - Líneas futuras.** Se muestra una serie de posibles mejoras susceptibles de ser implementadas por un futuro desarrollador que quisiera ampliar el alcance del proyecto.

- **Capítulo 12 - Bibliografía.** Se exponen las fuentes de las cuales se ha obtenido parte de la información y de las figuras utilizadas a lo largo del presente documento.

2 - ESTADO DE LA CUESTIÓN

A continuación, se muestra un conjunto de informaciones cuyo objetivo es aportar un marco que ayude a conocer, con mayor detalle, cada uno de los elementos relevantes en este proyecto.

2.1 - HISTORIA DE LA ROBÓTICA

Utilizaremos la definición de robot incluida en la 22ª edición del diccionario de la Real Academia Española.

- Robot. Máquina o ingenio electrónico programable, capaz de manipular objetos y realizar operaciones antes reservadas solo a las personas. [1]

Aunque es posible encontrar ejemplos de figuras capaces de manipular objetos y de realizar movimientos de forma autónoma ya en las antiguas civilizaciones griega y egipcia, consideraremos que estos elementos no reunían todos los requisitos necesarios para ser considerados robots según la definición anterior. [2]

Dadas las anteriores consideraciones, fijaremos el inicio del desarrollo de la robótica actual a comienzos del siglo XIX, cuando el francés Joseph Jacquard creó el telar programable que se muestra en la figura 2.1. Este telar actuaba siguiendo una serie de órdenes que recibía por medio de un conjunto de tarjetas de cartón perforadas. [3]

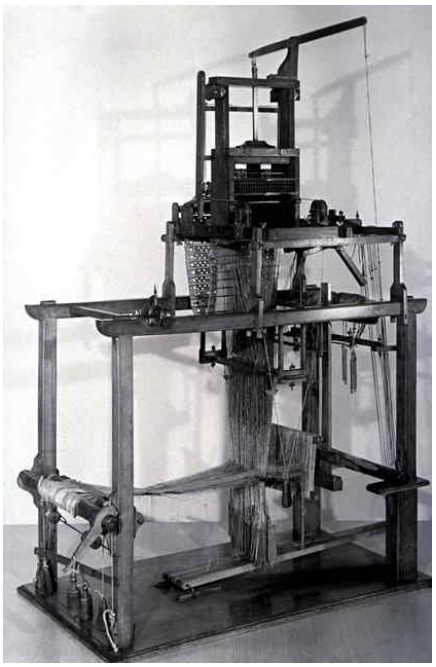


Figura 2.1: Telar de Jacquard [16]

Este desarrollo continuó durante todo el siglo XIX hasta que, ya en el siglo XX encontramos los primeros robots industriales, *los Unimate*s, desarrollados por George Devol y Joseph Engelberger, siendo considerado este último el padre de la robótica. En la figura 2.2 se muestra uno de estos robots *Unimate*. [2]

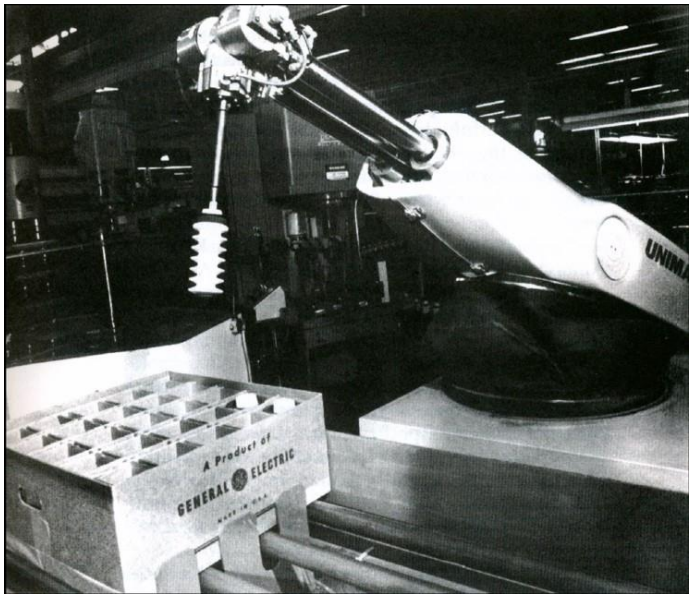


Figura 2.2: Robot *Unimate* [17]

La comercialización de robots comenzó en los años 50 y, en el año 1962, *General Motors* incluyó el primer robot industrial en su línea de producción de Trenton. Años más tarde, en 1970, se construyó, con propósitos de investigación, el conocido como *Brazo de Standford* el cual se muestra en la figura 2.3. [2]

Figura 2.3: Brazo de *Stanford* [18]

En el año 1978, la empresa *Unimation* desarrolló el robot *PUMA* (Programmable Universal Manipulator for Assembly) que se muestra en la figura 2.4 y en cuyas articulaciones es posible encontrar ya una semejanza directa con un brazo humano. [2]

Figura 2.4: Robot *PUMA* [19]

En el ámbito de los robots antropomórficos, es destacable la evolución de los robots *Honda*. En el año 1986, esta empresa desarrolló el *E0*, un robot bípedo carente de torso y brazos cuya función básica era la de caminar. La evolución de este robot E continuó hasta el año 1993 cuando se desarrolló el *E6*. En la figura 2.5 se muestran los diversos modelos creados durante este período. [4]

Figura 2.5: Robots *E* [4]

En el año 1993, *Honda* desarrolló un nuevo robot, el *P1*, que conservaba las características del anterior *E6* pero incorporaba, además, torso y brazos lo cual le permitía realizar nuevas tareas como, por ejemplo, empujar y transportar objetos. La evolución de este diseño continuó hasta el desarrollo del *P3* en el año 1997. En la figura 2.6 se muestran los robots construidos durante esta etapa. [4]

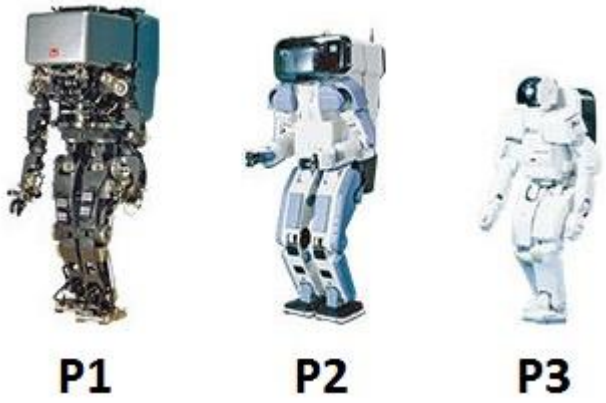


Figura 2.6: Robots *P* [4]

La evolución de los robots humanoides creados por *Honda* se completa con la aparición, en el año 2000, del robot *ASIMO*, mostrado en la figura 2.7, cuyo desarrollo continúa en la actualidad. [4]

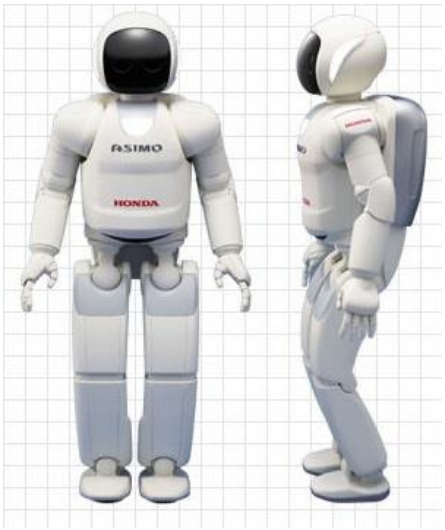


Figura 2.7: Robot *ASIMO 2011* [4]

En el año 2005, se formó la empresa *Aldebaran* que, un año más tarde, desarrollaría el primer prototipo del robot humanoide *NAO*. Desde entonces, este robot se ha convertido en una de las herramientas predilectas para investigadores y educadores y, en el año 2008, fue seleccionado como el sucesor del robot *Sony AIBO* en la *RoboCup Soccer League*. [5]

A diferencia de los robots PUMA, programados para realizar siempre la misma tarea, o los ASIMO, teleoperados, los robots NAO habitualmente son programados para actuar de forma autónoma.

En la figura 2.8 se muestra el robot NAO utilizado durante el desarrollo de este proyecto.

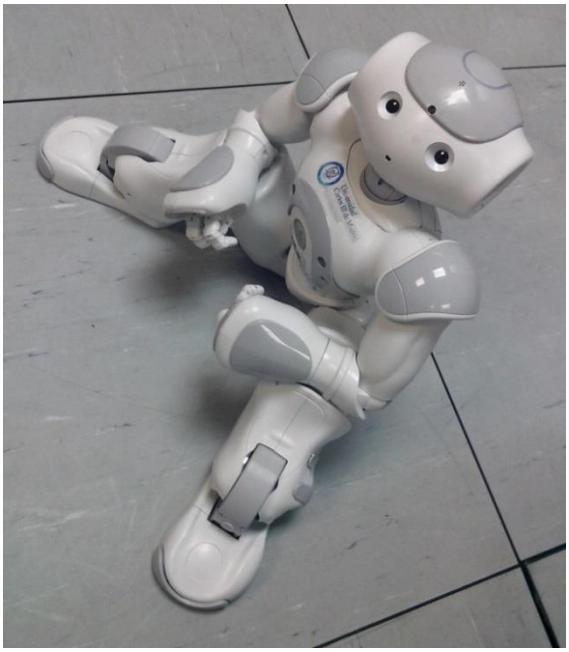


Figura 2.8: Robot *NAO*

2.2 - ROBOT HUMANOIDE *NAO*

El robot humanoide *NAO* es desarrollado por la empresa *Aldebaran* en el año 2006. Su objetivo final es convertirse en un miembro más de la familia siendo capaz de interactuar verbalmente con las personas de su entorno, ayudarles a realizar sus tareas o, incluso, reconocer sus caras y su estado de ánimo. [6]

Las dimensiones de este robot, mostradas en la figura 2.9, así como su reducido peso, en torno a los 5kg, permiten que se pueda trabajar en la programación con *NAO* de manera sencilla ya que puede ser transportado sin gran dificultad.

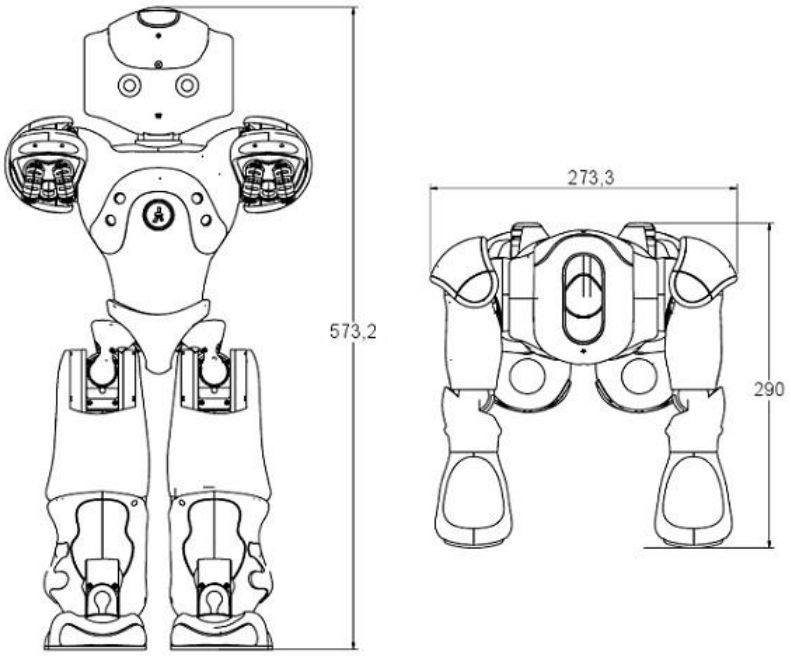


Figura 2.9: Dimensiones, en milímetros, del robot *NAO* [20]

Además de su aspecto humanoide, el robot *NAO* dispone del conjunto de articulaciones mostrado en la figura 2.10 que le permite la realización de una serie de movimientos muy similares a los de un ser humano.

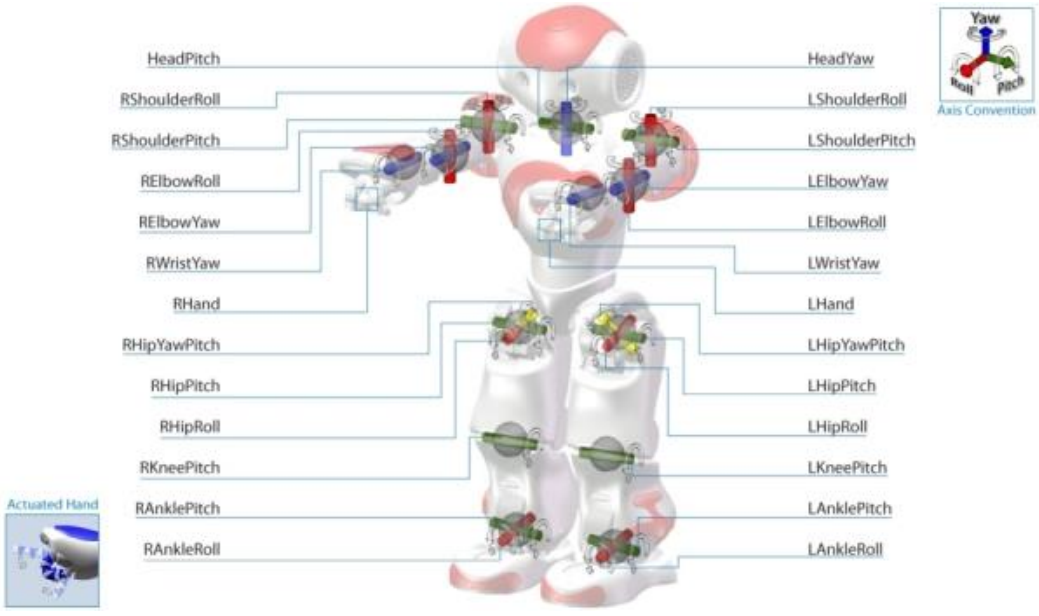


Figura 2.10: Articulaciones del robot *NAO* [21]

Además de estas articulaciones motorizadas, para una completa interacción con su entorno, el robot *NAO* dispone de un amplio conjunto de sensores y actuadores cuyos elementos más destacados se encuentran representados en la figura 2.11.

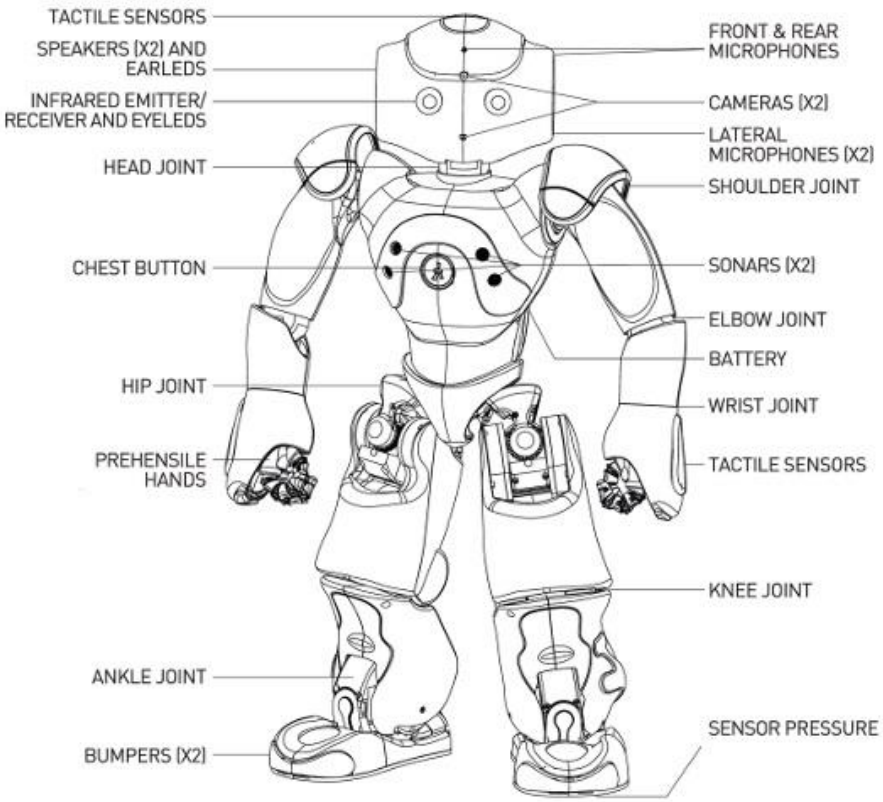


Figura 2.11: Posición de algunos de los sensores del robot *NAO* [22]

El robot puede conectarse a un ordenador por medio de una conexión *Ethernet* o de forma inalámbrica mediante una conexión *WiFi* y, además, puede ejecutar aplicaciones de forma autónoma gracias a su sistema operativo *NAOqi OS*. [7]

El desarrollo de aplicaciones se realiza a través del entorno *NAOqi* que contiene los diversos módulos necesarios para la utilización de las funciones requeridas para cada acción. Para facilitar este desarrollo, está disponible una Interfaz de Programación de Aplicaciones (API) cuyo propósito es aportar información detallada sobre las bibliotecas de funciones predeterminadas. Con los lenguajes de programación *C++* y *Python*, esta API puede ser utilizada de forma directa. [7]

2.3 - *CHOREGRAPHE*

Para facilitar la creación de nuevas aplicaciones, *Aldebaran* ha desarrollado también un simulador que permite la visualización de una imagen tridimensional de un robot capaz de ejecutar aplicaciones con el fin de servir como test de las mismas. Además de realizar test de código, *Choregraphe* permite la creación del mismo mediante la conexión de diversos elementos "box", cada uno de los cuales corresponde a una acción concreta a realizar. En la figura 2.12 se muestra la interfaz de *Choregraphe*.

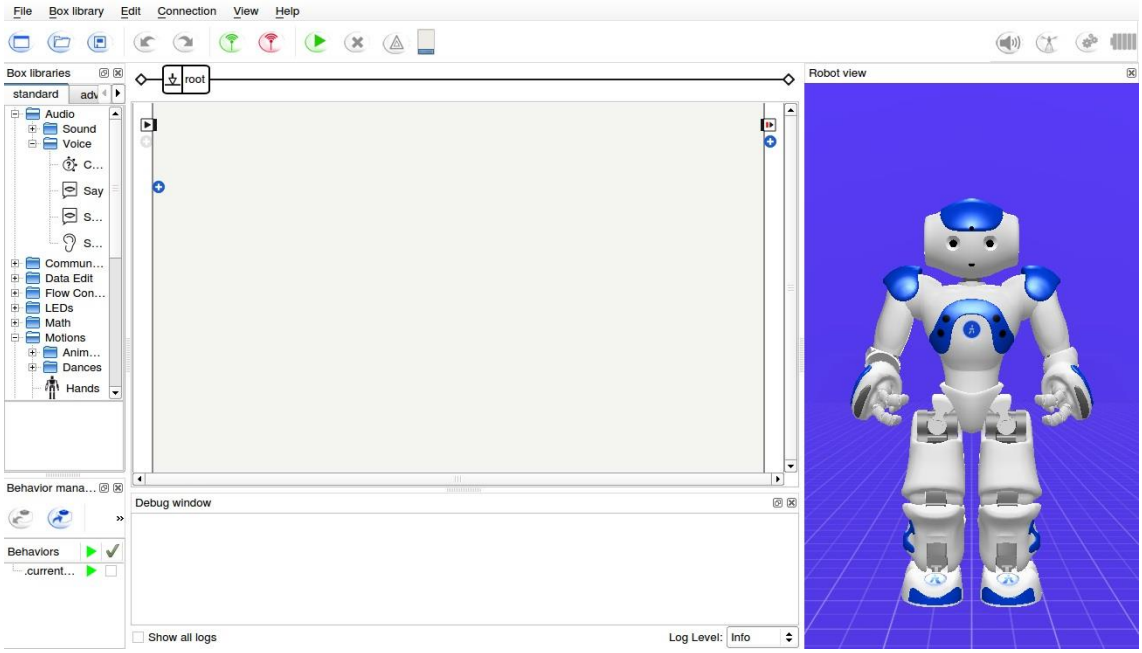


Figura 2.12: Interfaz de *Choregraphe* [23]

En caso de disponer de un robot real conectado al ordenador, *Choregraphe* permite su monitorización al sustituirlo por la imagen simulada. De este modo, es posible visualizar en pantalla el posicionamiento en tiempo real del NAO auténtico así como realizar captaciones de los ángulos de cada una de sus articulaciones.

2.4 - *PYTHON*

Python es un lenguaje de programación multiplataforma creado por Guido van Rossum en el año 1991 y distribuido bajo una licencia abierta, es decir, puede ser utilizado y editado de forma libre y sin coste alguno. [8]

Además de permitir una programación orientada a objetos con una sintaxis sencilla, posee una amplia librería de funciones estándar fácilmente ampliable mediante la elaboración de nuevos módulos que pueden ser utilizados indistintamente por cualquier código *Python*. [8] [9]

En la figura 2.13 se muestra el código de la declaración de una variable y una función extraído de la aplicación desarrollada para el robot *NAO* a lo largo de este proyecto. Al llamar a esta función, denominada "cambio", se modifica el valor de la variable "clave" asignándole el valor "nueva", el cual se corresponde al texto introducido entre paréntesis en el momento de la llamada.

```
clave = "espera"
def cambio(nueva):
        global clave
        clave = nueva
```

Figura 2.13: Ejemplo de declaración de una variable y de una función en *Python*

2.5 - *MUSICXML*

MusicXML es un formato basado en *XML* cuyo objetivo es permitir el almacenamiento e intercambio de partituras. [10]

XML permite almacenar información clasificándola mediante etiquetas para, de esta forma, facilitar su posterior acceso y recuperación. Estas etiquetas no están predefinidas por lo que el usuario puede crearlas y administrarlas según sus propias necesidades. En la figura 2.14 se muestra un ejemplo de documento escrito en formato *XML*. En este ejemplo se representa una "caja" que se encuentra almacenada en el "Almacén1" y contiene "Objetos". [11]

```
<caja>
        <almacenado>Almacén1</almacenado>
        <contenido>Objetos</contenido>
</caja>
```

Figura 2.14: Ejemplo de código *XML*

MusicXML se basa en una amplia variedad de etiquetas predeterminadas capaces de proporcionar información acerca de cada uno de los aspectos musicales de la partitura, así como de precisar algunos detalles relativos a la representación gráfica de algunos de sus elementos. En la figura 2.15 se muestra un ejemplo de utilización de *MusicXML* junto a su representación gráfica. [12]

Partitura	Código *MusicXML*
	<?xml version="1.0" encoding="UTF-8" standalone="no"?> <!DOCTYPE score-partwise PUBLIC "-//Recordare//DTD MusicXML 3.0 Partwise//EN" "http://www.musicxml.org/dtds/partwise.dtd"> <score-partwise version="3.0"> <part-list> <score-part id="P1"> <part-name>Music</part-name> </score-part> </part-list> <part id="P1"> <measure number="1"> <attributes> <divisions>1</divisions> <key> <fifths>0</fifths> </key> <time> <beats>4</beats> <beat-type>4</beat-type> </time> <clef> <sign>G</sign> <line>2</line> </clef> </attributes> <note> <pitch> <step>C</step> <octave>4</octave> </pitch> <duration>4</duration> <type>whole</type> </note> </measure> </part> </score-partwise>

Figura 2.15: Ejemplo de código *MusicXML* [24]

El uso de este estándar es muy extendido y, actualmente, existen un gran número de programas con capacidad de lectura y escritura de archivos en formato *MusicXML* lo que permite simplificar la creación de partituras propias así como la búsqueda de melodías creadas por otros usuarios. [10]

2.6 - *MUSESCORE*

Se trata de un software libre y multiplataforma que permite la elaboración de partituras mediante un editor *WYSIWYG* (acrónimo en inglés de "lo que ves es lo que obtienes"), es decir, permite trabajar directamente sobre un pentagrama introduciendo, de forma sencilla y directa, los elementos necesarios para elaborar la melodía deseada tal y como se muestra en la figura 2.16. [13]

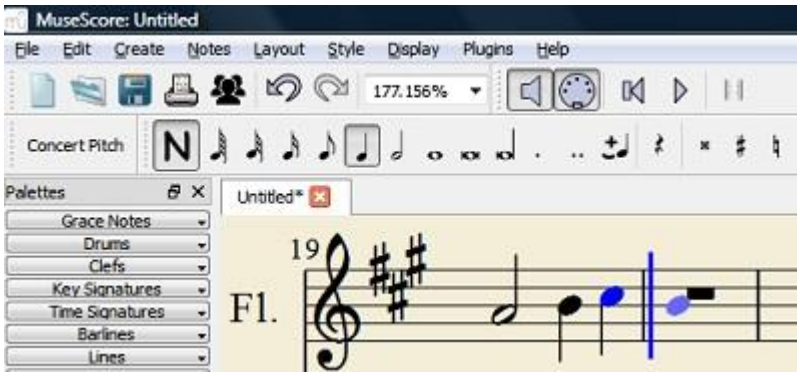


Figura 2.16: Introducción de notas en una partitura con *MuseScore* [13]

Como complemento del editor de partitura, *MuseScore* cuenta con un reproductor capaz de mostrar, en cualquier momento, el resultado del trabajo realizado aportando, de este modo, un gran apoyo al usuario durante la fase de desarrollo al permitirle escuchar la melodía elaborada. [13]

Además de todo esto, *MuseScore* está disponible en más de 40 idiomas y soporta los formatos *MIDI* y *MusicXML*, dos de los formatos más extendidos para el intercambio de melodías. [13]

2.7 - OTROS PROYECTOS

En este proyecto, se buscaba el desarrollo de una aplicación capaz de crear algún tipo de entretenimiento por lo que, desde un primer momento, se buscaron proyectos relacionados con la música como fuente de inspiración.

Al buscar en la web, se encontró que ya existían numerosos proyectos en los que el robot *NAO* bailaba al ritmo de una melodía por lo que se decidió que, para desarrollar una aplicación más novedosa, el robot participaría activamente en la creación de la melodía, es decir, sería uno de los músicos.

Relacionado con esta aplicación, se encontró la publicación "Using Visual and Auditory Feedback for Instrument-Playing Humanoids" relativa a un proyecto desarrollado en la universidad de Freiburg en el cual se logra que un robot *NAO*, gracias a la utilización de un *feedback* auditivo y de la visión ofrecida por sus cámaras, interprete melodías de forma correcta y eficiente mediante la utilización de un metalófono de 11 láminas y 2 baquetas adaptadas a las manos del robot. [14]

Con el objetivo de diferenciarse de los anteriores proyectos y de contribuir a la creación de nuevas funcionalidades para el robot *NAO*, se decidió que se utilizaría un instrumento que no perteneciera a la familia de instrumentos de percusión.

Ampliando la búsqueda más allá de los robots *NAO*, se encontró el robot violinista de *Toyota* que se muestra en la figura 2.17. Este robot, además de otras funcionalidades, es capaz de interpretar melodías con un violín utilizando sus propias manos. [15]

Figura 2.17: Robot violinista creado por *Toyota* [15]

Tras visualizar el resultado del proyecto de *Toyota*, se decidió que, en la nueva aplicación, el robot *NAO* utilizaría sus propias manos para interpretar la melodía ya que se consideró que la interacción directa robot-instrumento mejoraría el aspecto visual de la acción.

Dado que las limitaciones del propio robot dificultaban en gran medida la utilización de instrumentos de cuerda pulsada o frotada, tales como guitarra o violín, se decidió finalmente que el instrumento a utilizar sería un teclado electrónico con un número de teclas superior al de las láminas del metalófono utilizado en la universidad de Freiburg para, de este modo, ampliar el rango de melodías interpretables.

3 - OBJETIVOS

El objetivo de este proyecto es construir un sistema que logre generar entretenimiento por medio del uso de un robot *NAO* capaz de tocar un instrumento musical que, en este caso, es un teclado electrónico de tres escalas.

Es necesario que el sistema cumpla un conjunto de requisitos:

- El robot debe ser capaz de reproducir las partituras que le sean introducidas en un formato estandarizado para facilitar la obtención de nuevas partituras por parte del usuario.

- El formato escogido para las partituras debe ser, a su vez, uno de los formatos de salida de algún software que permita elaborarlas de forma sencilla o leerlas en otros formatos para su conversión. De esta forma, el usuario no necesitará disponer de un amplio conocimiento en el formato seleccionado para poder crear sus propias melodías.

- La melodía interpretada debe ser reconocible por el usuario, es decir, el tiempo en que suena cada una de las notas tiene que estar directamente relacionado con las figuras musicales encontradas en la partitura.

- El sistema debe poder utilizarse de forma sencilla y, para facilitar que el usuario pueda acompañar al robot con su propio instrumento musical, la interacción usuario-robot se realizará sin contacto físico entre ambos.

- El teclado debe disponer de un soporte propio que permita situarlo a una altura determinada que sea constante a lo largo de múltiples utilizaciones de la aplicación.

- La aplicación debe elaborarse en un formato que permita su ejecución de forma automática en el robot, es decir, al encender el robot éste comenzará la acción sin necesidad de utilizar un ordenador para lanzar la aplicación.

- El usuario debe tener la posibilidad de escoger la partitura a reproducir de una lista introducida previamente en el robot. Además, dicha lista se podrá gestionar de manera sencilla por el usuario sin importar los conocimientos informáticos del mismo.

- Para poder realizar una función didáctica, el robot debe proporcionar información relativa a la nota interpretada en cada momento. Además, esta información debe ser proporcionada de manera que el usuario no necesite encontrarse en un punto determinado frente al robot.

- A pesar de la precisión requerida para este proyecto, el teclado podrá ser posicionado frente al robot de forma rápida y sencilla para maximizar el tiempo real de uso de la aplicación mediante la reducción del tiempo de preparación necesario incrementando, además, la eficacia de la aplicación al minimizar las probabilidades de un posicionamiento erróneo.

4 - CARACTERÍSTICAS DE LOS COMPLEMENTOS

La aplicación, para desarrollar la totalidad de sus funcionalidades, requiere que el robot disponga de un complemento, en este caso un teclado, con el que pueda interaccionar.

Dado que no se han incorporado funciones para detectar la posición del teclado, era necesario que éste se encontrara en una ubicación determinada, es decir, que tanto la distancia al robot como la altura sobre el suelo tuvieran un valor prefijado, que permaneciera invariable, con el objeto de garantizar el correcto funcionamiento del sistema en cada una de sus utilizaciones. Para lograr este objetivo, se fabricó un soporte que situara el teclado en una posición que facilitara el acceso al mismo ya que, en caso de situar el instrumento directamente sobre el suelo, las piernas del robot limitarían el movimiento de sus brazos, impidiéndoles alcanzar algunas de las teclas.

Tras estas consideraciones, se concluyó que los complementos necesarios para elaborar y, posteriormente, utilizar la aplicación serían los siguientes:

- Teclado
- Soporte

4.1 - TECLADO

Puesto que se pretendía que la interacción con el instrumento se realizara sin que fuera necesario el uso de ningún accesorio adicional, era requisito indispensable que el ancho de tecla fuera superior al de los dedos del robot.

En la figura 4.1 se muestra la mano del robot *NAO* en su posición abierta. Esta disposición de los dedos, dada la invariabilidad de las distancias entre los mismos, dificulta en gran medida la pulsación de varias teclas, de forma simultánea, con una sola mano. Por esta razón, el robot únicamente podría pulsar, por medio de la utilización coordinada de ambos brazos, un máximo de dos teclas al mismo tiempo, es decir, el teclado debía tener la capacidad de producir, al menos, dos sonidos de manera simultánea para no limitar las posibilidades de uso de las funcionalidades incorporadas en la aplicación.

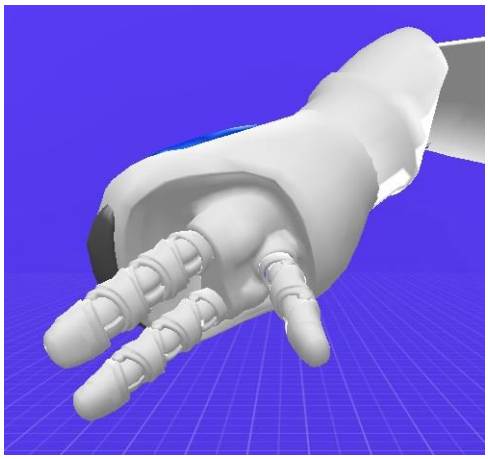


Figura 4.1: Mano derecha del robot *NAO* [19]

Para este proyecto, se ha seleccionado el teclado electrónico de la marca "PLAY ON" que se muestra, con sus medidas más relevantes, en la figura 4.2 puesto que, con un bajo coste económico, cumple los requisitos mínimos mencionados anteriormente. Además, sus tres escalas permiten la interpretación de una gran variedad de melodías dado que la envergadura del robot es suficiente para alcanzar la totalidad del teclado. Cabe destacar que, aunque el teclado dispone de un total de 37 teclas, solamente se utilizarán las 22 blancas ya que la forma del puño del robot NAO dificulta el acceso a las negras que, además, son más pequeñas y requieren de una mayor presión para poder ser accionadas correctamente. Aunque la aplicación se ha desarrollado específicamente para su uso con este teclado, es válida también para cualquier otro teclado con el mismo ancho de tecla y un número de teclas blancas igual o inferior.

Figura 4.2: Teclado con sus medidas más relevantes

Adicionalmente, este teclado permite seleccionar, de entre una lista de 8 opciones que imitan el timbre de otros instrumentos musicales, el tipo de sonido producido y, tras pulsar cualquiera de las teclas, un conjunto de LEDs rojos se iluminan de manera intermitente bajo los protectores azules situados alrededor de cada uno de los altavoces. Todo esto supone una mejora en la estética del proyecto, haciéndolo más atractivo para el usuario final.

4.2 - SOPORTE

Tras varias pruebas en el laboratorio, se concluyó que la ubicación óptima para facilitar el correcto pulsado de las teclas se encontraba a una altura similar a la de los hombros del robot, puesto que ésta permitía el uso de una mayor superficie del dedo, mejorando así la precisión de las acciones al aumentar la probabilidad de que la pulsación se realice. Dado este requerimiento dimensional, se decidió que la solución más conveniente sería la fabricación de una mesa diseñada específicamente para su uso en este proyecto.

Tras haberse decidido que la posición del robot NAO durante la acción sería similar a su postura predeterminada “Crouch”, se concluyó que la altura óptima del soporte sería de, aproximadamente, 27.5 cm.

El único elemento a soportar sería el teclado por lo que las dimensiones del soporte, en el plano horizontal, debían ser ligeramente superiores a las del propio teclado para poder aprovechar los pequeños elementos antideslizantes incorporados en el mismo. De este modo, es posible prescindir de cualquier tipo de fijación adicional entre teclado y soporte.

Con todo esto, se realizó el diseño previo que se muestra en la figura 4.3. Dado que toda la presión sería ejercida sobre las teclas en la parte delantera de la mesa, el hecho de situar más próximas entre sí las patas traseras no comprometía la estabilidad del soporte durante la acción. Esta aproximación facilitaría una posible unión de estas patas a un elemento externo como, por ejemplo, un asiento diseñado para el robot al ser la distancia entre ellas similar a la anchura del propio robot *NAO*.

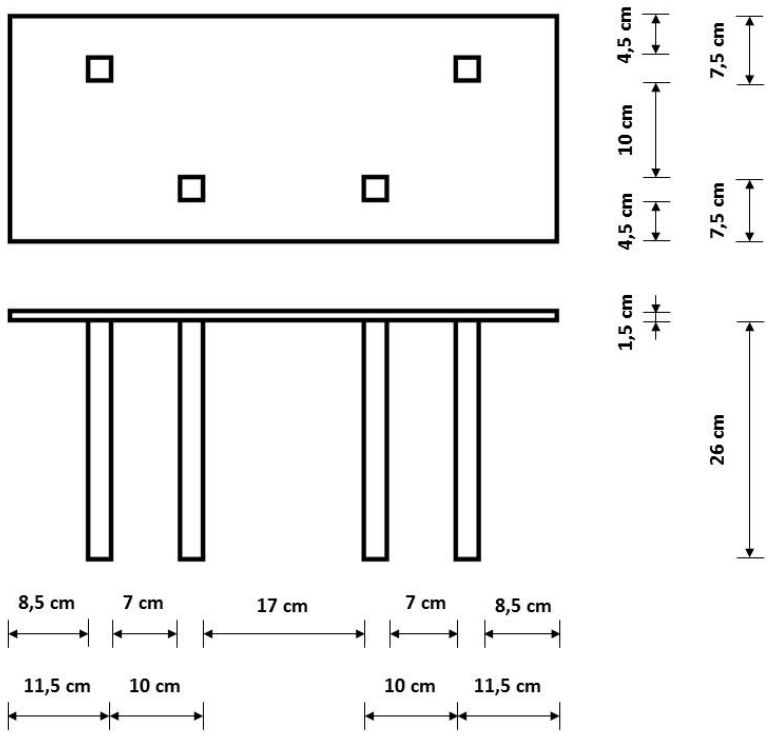


Figura 4.3: Planos acotados del soporte

En cuanto a los materiales seleccionados, se decidió que el soporte estaría compuesto por un tablero de aglomerado de 1.5 cm de espesor montado sobre 4 listones de madera de 3 por 3 cm. Estos elementos, además de suponer un coste económico no muy elevado, garantizan en gran medida la solidez del soporte y la estabilidad de sus dimensiones.

Para reducir el espacio ocupado por el soporte mientras no está siendo utilizado, se decidió que sus patas serían diseñadas de tal forma que pudieran ser extraídas con facilidad. De este modo se conseguía además que, en caso de desarrollarse aplicaciones adicionales en las que el robot utilizara diferentes posturas, únicamente fuera necesario elaborar nuevos juegos de patas e intercambiarlos con el original sin que esto produjera ningún deterioro en el soporte.

En la figura 4.4 se muestra el sistema de fijación diseñado para unir las patas al tablero. Este sistema se compone de dos escuadras con agujeros de 4 mm atornilladas al aglomerado una frente a la otra a una distancia de 3 cm, un listón con un agujero pasante también de 4 mm apoyado en el tablero entre las dos escuadras mencionadas y un tornillo roscado que completa la sujeción atravesando todos los elementos anteriores. El tornillo se fija por medio de una mariposa para que el sistema pueda montarse y desmontarse de forma rápida y sencilla, sin necesidad de disponer de herramientas adicionales.

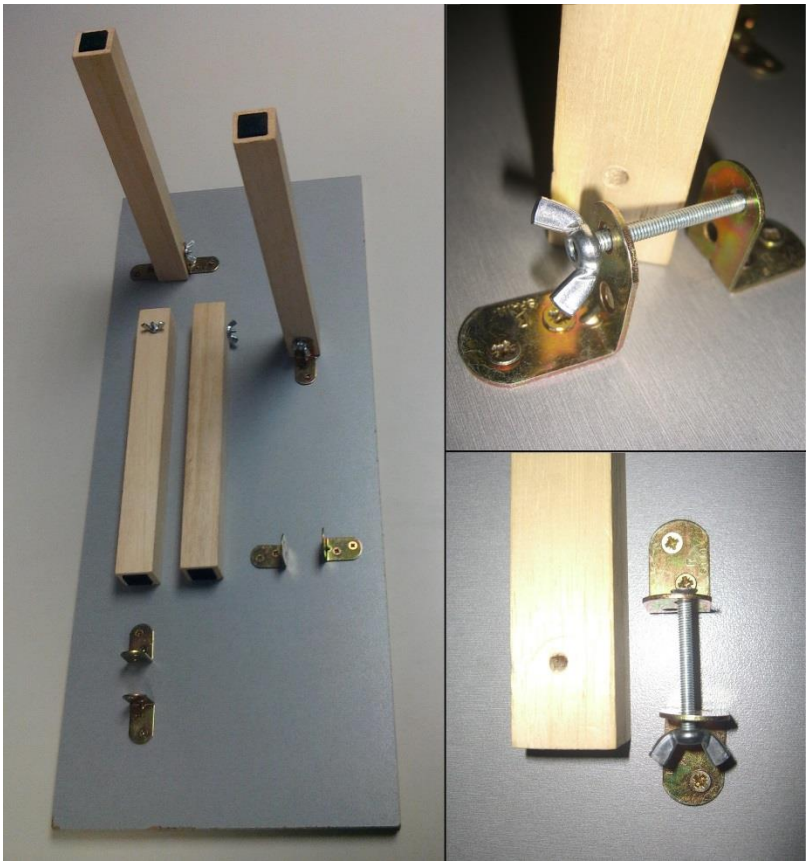


Figura 4.4: Detalle de la fijación de las patas del soporte

5 - DISEÑO Y DESARROLLO DE LA APLICACIÓN

5.1 - ARQUITECTURA DE LA APLICACIÓN

En la figura 5.1 se muestra un gráfico descriptivo de la arquitectura de la aplicación y, a continuación, se incluye una descripción de alto nivel de cada uno de los módulos que la componen.

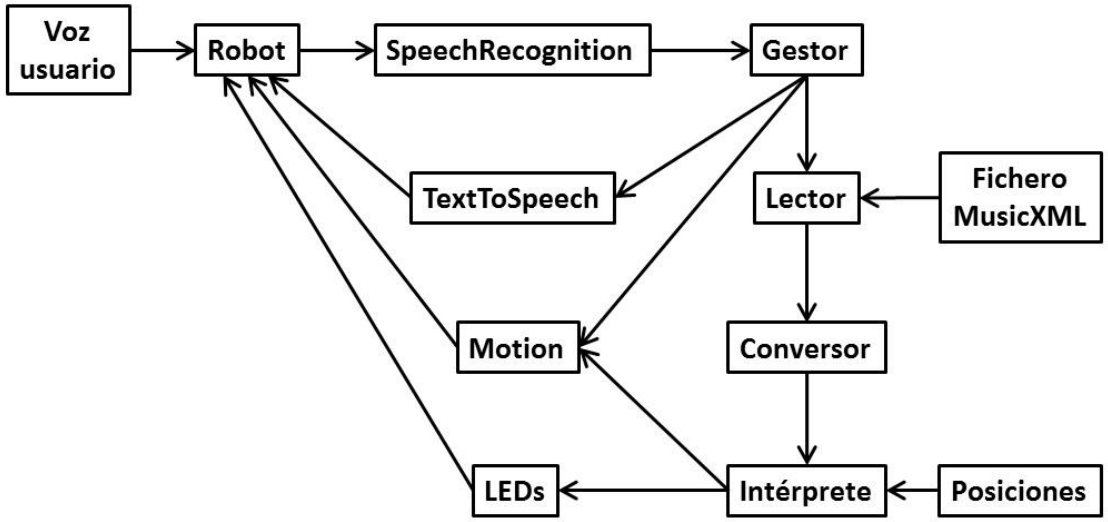


Figura 5.1: Arquitectura de la aplicación

Voz usuario. Conjunto de las instrucciones de voz utilizadas por el usuario para controlar el funcionamiento del sistema.

Robot. Robot humanoide NAO encargado de recibir las instrucciones de voz del usuario y de realizar las acciones que sean requeridas por la aplicación para llevar a cabo la acción solicitada.

SpeechRecognition. Módulo interno del robot encargado de generar los eventos relacionados con el reconocimiento de las palabras previamente escogidas como instrucciones válidas.

TextToSpeech. Módulo interno del robot encargado de conseguir que éste recite el texto proporcionado por la aplicación.

Motion. Módulo interno del robot encargado de lograr que el mismo realice los movimientos requeridos por la aplicación.

Gestor. Módulo principal de la aplicación puesto que es el encargado de gestionar la interacción con el usuario. Este módulo, en función de la información recibida desde SpeechRecognition, toma las decisiones de enviar una secuencia de movimientos a Motion, un texto a TextToSpeech o el identificativo de una melodía solicitada por el usuario al módulo lector.

Fichero MusicXML. Archivo de partitura importado por la aplicación desde el directorio "Partituras".

Lector. Módulo de la aplicación encargado de la lectura de la partitura indicada por el módulo gestor. Importa el fichero *MusicXML* desde la carpeta "Partituras" y recopila la información necesaria para que el robot interprete la melodía para, posteriormente, enviar estos datos al módulo conversor.

Conversor. Módulo de la aplicación encargado de la conversión de los datos aportados por el módulo lector a un formato más sencillo para su interpretación.

Posiciones. Datos referentes a las posiciones relativas de las teclas que permiten el acceso a las mismas por parte del robot.

Leds. Módulo interno del robot encargado del control de la iluminación de cada uno de los LEDs incorporados en el mismo.

Intérprete. Módulo de la aplicación encargado de interpretar los datos recibidos desde el módulo conversor para, posteriormente, enviar las posiciones adecuadas al robot a través del módulo Motion y proporcionar, al módulo Leds, la información necesaria para que los LEDs actúen en sincronía con cada uno de los movimientos realizados.

5.2 - DIAGRAMA DE FLUJO DE LA APLICACIÓN

En la figura 5.2 se expone el diagrama de flujo correspondiente a la aplicación desarrollada a lo largo de este proyecto. Este diagrama es una completa referencia de las acciones que el robot debe realizar a lo largo de la ejecución del código.

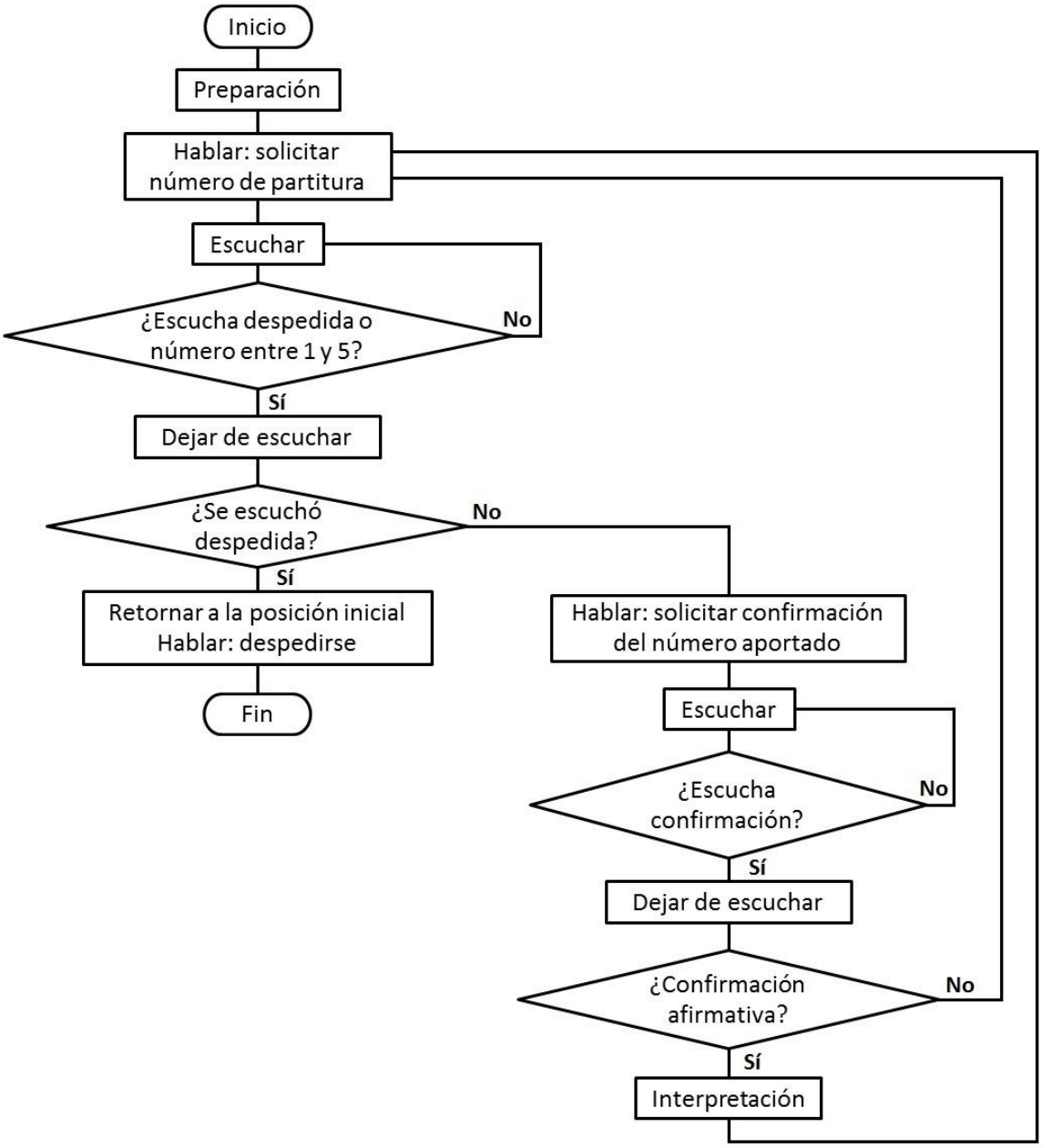


Figura 5.2: Diagrama de flujo de la aplicación

Las acciones “Preparación” e “Interpretación” poseen diagramas de flujo propios que han sido extraídos del principal con el objeto de simplificar la lectura de los mismos. Estos diagramas adicionales se muestran en las figuras 5.3 y 5.4 respectivamente.

Figura 5.3: Diagrama de flujo de la acción "Preparación"

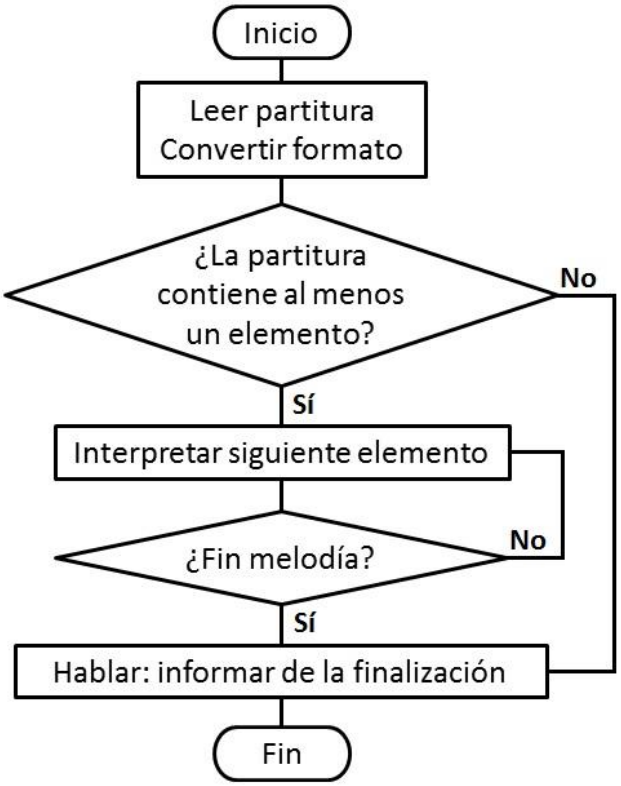


Figura 5.4: Diagrama de flujo de la acción "Interpretación"

5.3 - PROCESO DE DESARROLLO

Para la elaboración de este proyecto, se ha utilizado un sistema *Linux*, concretamente la versión 1.12 de *Ubuntu*, con el software libre *MuseScore* y las versiones 2.7.3 de *Python*, 1.12.3 de *NAOqi* y 1.12.3 de *Choregraphe*.

En lo relativo a la metodología utilizada para elaborar el *script*, se ha optado por la adición de nuevas funcionalidades a una aplicación base que permitiera, desde su primera versión, realizar las acciones elementales de leer partituras y reproducir melodías. Este proceso ha atravesado las siguientes etapas:

- Elaboración de la aplicación base
- Captación de las posiciones relativas de las teclas
- Selección del formato de las partituras
- Utilización de los LEDs faciales
- Comunicación verbal y órdenes por voz
- Pruebas finales con captación de temperaturas

5.3.1 - ELABORACIÓN DE LA APLICACIÓN BASE

Las primeras partituras introducidas a la aplicación se elaboraron como ficheros de texto con un formato simplificado y adaptado a las características del teclado escogido. Este archivo *txt* debía contener en cada línea la letra asignada a la tecla que se debía tocar seguida de un número que correspondía a la duración marcada por la figura musical correspondiente a dicha nota o, en caso de que la melodía hubiera finalizado, un corchete indicativo de dicho fin.

Para poder utilizar el teclado, el robot debía encontrarse en una posición que le permitiera presionar con la fuerza necesaria sin que existiera riesgo de caída. Se escogió, de la librería de posturas predeterminadas para el robot NAO, la postura “Crouch”, que se muestra en la figura 5.5, por tratarse de la más adecuada a nivel estético. Sobre esta posición predeterminada, se realizaron pequeñas modificaciones orientadas a mejorar la estabilidad durante la acción.

Figura 5.5: Postura “Crouch”

Una vez alcanzada la postura estable, era necesaria la ejecución de una secuencia de movimientos que posicionara las manos del robot sobre el teclado sin golpearlo. La elaboración de dicha secuencia se ha realizado mediante el empleo, en *Choregraphe*, de un “box” de tipo “timeline”, que permite la captación del ángulo de cada articulación del robot en intervalos de tiempo regulares para, una vez finalizada la animación creada al mover el programador las extremidades del robot, modificarla y simplificarla como se muestra en la figura 5.6. Una vez finalizada la edición, es posible exportar el resultado como un *script* de *Python* obteniendo de esta forma el código que debe ser ejecutado para la realización de la acción deseada.

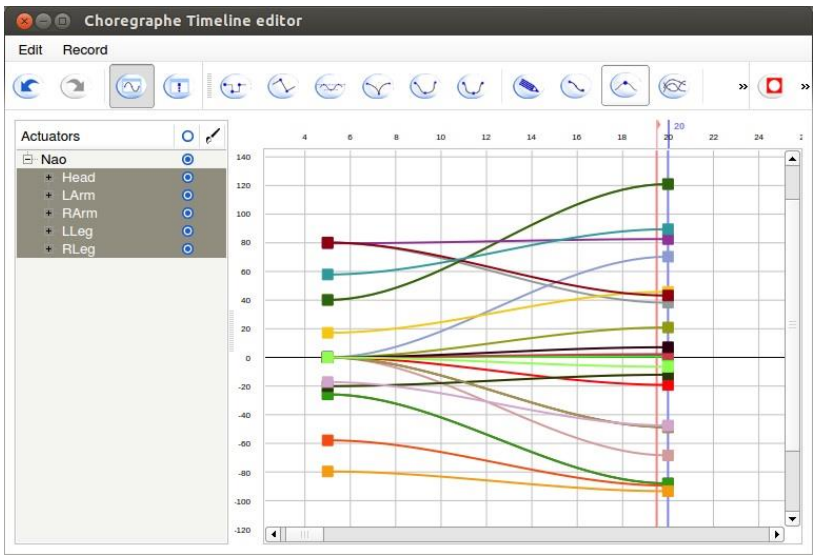


Figura 5.6: Ejemplo de edición de animación en *Choregraphe* [23]

Siguiendo el mismo procedimiento, se ha creado también una secuencia de movimientos que, una vez finalizada la melodía, retorna los brazos a su posición original para evitar la colisión de las manos con el teclado al retornar a la postura inicial en caso de una nueva ejecución de la aplicación, o a la postura “Crouch” en caso de producirse el apagado del robot.

Disponiendo de las partituras y de las animaciones anteriores y posteriores a la acción, ya era posible la elaboración de la aplicación base. Este *script* inicial, al ser ejecutado, posicionaba el robot, leía la partitura, esperaba el tiempo correspondiente a cada nota (sin acción del robot dado que aún no existen posiciones relativas a las teclas) y, finalmente, retornaba el robot a su posición inicial. En la figura 5.7, se muestra la posición utilizada para alcanzar las teclas.

Figura 5.7: Postura utilizada para alcanzar la teclas

5.3.2 - CAPTACIÓN DE LAS POSICIONES RELATIVAS DE LAS TECLAS

Para interpretar una nota, el robot sitúa el dedo sobre la tecla correspondiente, baja la mano una distancia suficiente para emitir el sonido, espera el tiempo marcado por el valor de la duración y vuelve a subir la mano para finalizar la acción. A cada brazo le corresponde tocar una mitad del teclado.

Dadas las características físicas del proyecto, se ha considerado que la forma más sencilla y fiable de alcanzar el objetivo es la captación, a través de *Choregraphe* y su "box" de tipo "timeline" mencionado en el apartado anterior, de cada una de las posiciones necesarias. Para llevar a cabo este proceso, se han almacenado todos los ángulos de las articulaciones del brazo correspondiente, como se muestra en la figura 5.8, estando éste físicamente en la posición deseada con el objeto de realizar la exportación a un *script* de *Python* que permitiera la inclusión de cada posición en la aplicación base. A diferencia de lo que ocurriera en el apartado anterior, al tratarse de posiciones únicas y no de una animación, no es necesario acceder a la pantalla de edición tras la captación de ángulos.

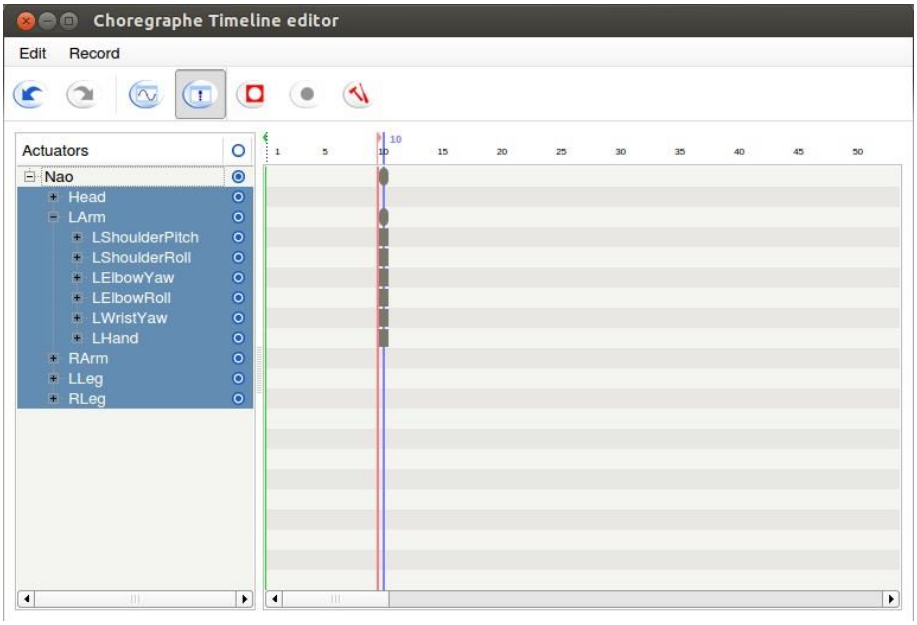


Figura 5.8: Captación de ángulos en *Choregraphe* [23]

Cada mano tiene capacidad para tocar una única tecla en cada aproximación pero es posible la interpretación de dos notas simultáneas siempre que cada una de ellas corresponda a un brazo diferente. Para permitir esta acción, se han creado nuevas posiciones correspondientes a pares de teclas combinando las ya disponibles tal y como se muestra en la figura 5.9. De esta forma, al requerirse la interpretación de dos notas de forma simultánea, cada mano se posicionaría sobre su tecla correspondiente y ambas descenderían para, posteriormente, ascender al mismo tiempo para lograr la creación de un único sonido combinado. Aunque las funciones utilizadas crean una secuencia y no una sincronización, la elevada velocidad de ejecución de la función impide que el usuario pueda percibir este hecho.

```
#Tocar tecla a (mapeada mesa)
aNamesB = list()
aKeysB = list()

aNamesB.append("LElbowRoll")
aKeysB.append(-0.61049)

aNamesB.append("LElbowYaw")
aKeysB.append(-0.32218)

aNamesB.append("LHand")
aKeysB.append(0.01745)

aNamesB.append("LShoulderPitch")
aKeysB.append(0.26994)

aNamesB.append("LShoulderRoll")
aKeysB.append(0.84366)

aNamesB.append("LWristYaw")
aKeysB.append(1.82387)
```

```
#Tocar tecla v (mapeada mesa)
vNamesB = list()
vKeysB = list()

vNamesB.append("RElbowRoll")
vKeysB.append(0.57683)

vNamesB.append("RElbowYaw")
vKeysB.append(0.24233)

vNamesB.append("RHand")
vKeysB.append(0.01745)

vNamesB.append("RShoulderPitch")
vKeysB.append(0.20253)

vNamesB.append("RShoulderRoll")
vKeysB.append(-0.79159)

vNamesB.append("RWristYaw")
vKeysB.append(-1.82387)
```

```
#Tocar teclas av
avNamesB = list()
avKeysB = list()

for i in aNamesB:
        avNamesB.append(i)
for i in aKeysB:
        avKeysB.append(i)
for i in vNamesB:
        avNamesB.append(i)
for i in vKeysB:
        avKeysB.append(i)
```

Figura 5.9: Creación de las posiciones “a”, “v” y “av”

Una vez finalizada la captación, para verificar la validez de las posiciones utilizadas por la aplicación, se han interpretado escalas completas tanto en sentido ascendente como descendente. Esta prueba ha sido necesaria debido a que, en el caso de pequeños desplazamientos, se generan imprecisiones considerables cuya relevancia depende también del sentido en el que se realice el movimiento. A pesar de la existencia de estos pequeños errores, no ha sido necesario ningún reajuste de las posiciones después de haber superado el mencionado test de las escalas.

5.3.3 - SELECCIÓN DEL FORMATO DE LAS PARTITURAS

Aunque el formato utilizado hasta este punto del proyecto permitía el buen funcionamiento de la aplicación, no era válido ya que, al no tratarse de un formato estandarizado y de uso común, dificultaba el uso de la aplicación por parte del usuario al no existir la posibilidad de encontrar partituras ya elaboradas.

Desde un primer momento, se creyó adecuado el uso de un formato basado en el lenguaje *XML* ya que su lectura resultaría más sencilla que la de otros formatos como, por ejemplo, *midi*. Finalmente, se escogió *MusicXML* por tratarse de un formato estandarizado y de uso extendido basado en el mencionado lenguaje *XML*.

Al realizar el cambio del formato del archivo leído por la aplicación, fue necesario modificar la misma incluyendo una función de conversión que, tras la lectura del archivo *MusicXML*, retornara una lista de notas convertidas al antiguo formato utilizado. La inclusión de esta nueva funcionalidad permitía utilizar la aplicación base sin realizar ningún cambio adicional a la modificación del código encargado de la lectura de partituras.

Con estas modificaciones, era posible encontrar partituras ya elaboradas que fueran válidas para la aplicación, pero su elaboración manual seguía presentando una complejidad elevada por lo que se decidió buscar un software de edición de partituras que tuviera, como formato de salida, *MusicXML*. Se decidió utilizar *MuseScore* ya que se trata de un software libre y multiplataforma que permite la edición de partituras de forma visual sobre un pentagrama tal y como se muestra en la figura 5.10 para, posteriormente, exportarlas en el formato deseado. Además de permitir la sencilla creación de partituras, este software también puede leer partituras en formato *MusicXml* y midi permitiendo así la modificación de partituras ya creadas y ampliando el número de archivos disponibles al poder crear ficheros *MusicXML* a partir de otros ficheros *midi*.

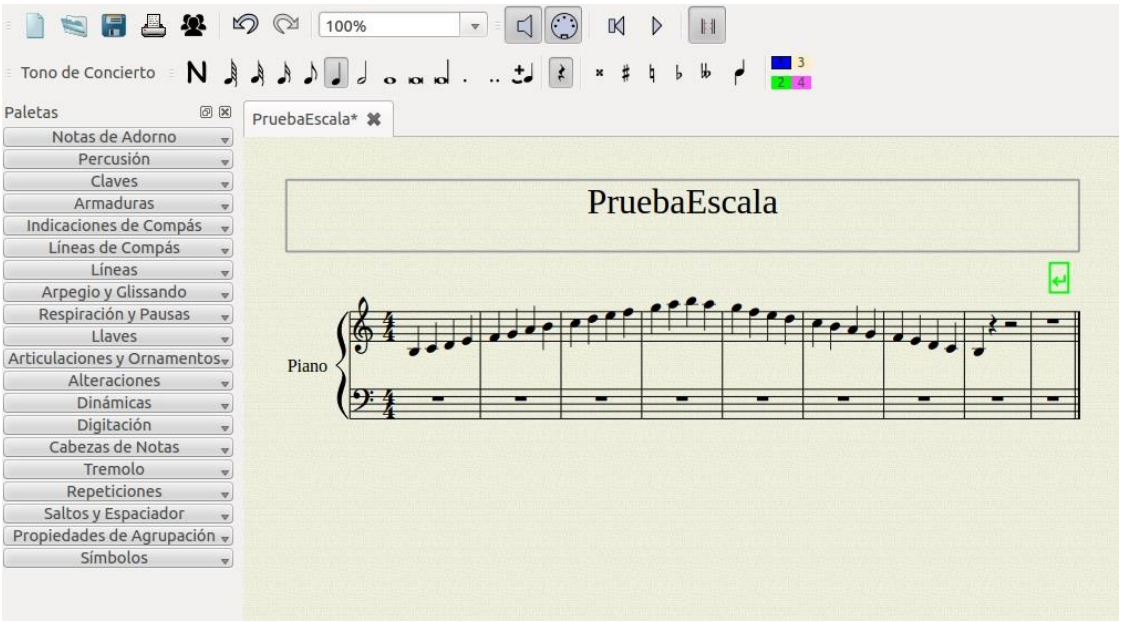


Figura 5.10: Escala creada en *MuseScore* [25]

Para facilitar también la adición de partituras susceptibles de ser leídas por la aplicación, se decidió que éstas se encontrarían dentro de una carpeta denominada “Partituras” que contendría, además, un fichero “index.txt” donde la aplicación buscaría el nombre del documento para, posteriormente, encontrarlo dentro de la carpeta. De esta forma, aunque todas las partituras se encuentren dentro del mismo directorio, la aplicación únicamente reconocerá aquellas que hayan sido correctamente identificadas en “index.txt” simplificando así la gestión de las melodías por parte del usuario.

5.3.4 - UTILIZACIÓN DE LOS LEDS FACIALES

Como funcionalidad adicional, se requería que el robot proporcionara información relativa a la nota que interpreta en cada momento. Con estos datos adicionales, el usuario puede conocer la tecla que se está pulsando sin necesidad de mirar el teclado consiguiendo, de esta forma, que resulte más sencillo imitar las acciones del robot en un segundo teclado o elaborar un acompañamiento con otro instrumento musical.

Se escogieron los LEDs faciales situados en los ojos del robot como elemento indicativo de la nota reproducida. Se seleccionó el color blanco para indicar los silencios así como un color diferente para cada una de las notas “Do”, “Re”, “Mi”, “Fa”, “Sol”, “La” y “Si” tal y como se muestra en la figura 5.11 siendo el color de cada nota idéntico para cada una de las escalas disponibles en el teclado.

R	G	B	Color	Nota Programa	Nota
FF	FF	FF		Silence	Silencio
FF	0	0		C	Do
FF	80	0		D	Re
FF	FF	0		E	Mi
0	FF	0		F	Fa
0	FF	FF		G	Sol
0	0	FF		A	La
80	0	FF		B	Si

Figura 5.11: Colores utilizados y su significado

El robot NAO dispone de un total de 8 LEDs en cada ojo que pueden ser utilizados de forma individual. Por este motivo, a modo de presentación de los colores que se utilizarán durante la acción y tras haberse realizado el posicionamiento inicial, la aplicación ilumina cada uno de estos LEDs, de forma secuencial y ordenada en ambos ojos, en cada uno de los 8 colores escogidos anteriormente. De esta forma, representada en la figura 5.12, se muestran las posibilidades cromáticas de esta funcionalidad de forma previa a su utilización.

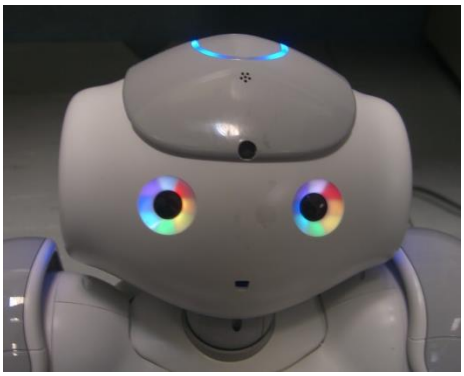


Figura 5.12: Exposición inicial de todos los colores

Durante la interpretación de una melodía, todos los elementos de un mismo ojo son tratados como una única entidad con el objetivo de hacer los colores más visibles mediante el incremento del área iluminada. En la figura 5.13, se muestra la diferencia entre la utilización de un único LED y los 8 de forma conjunta.

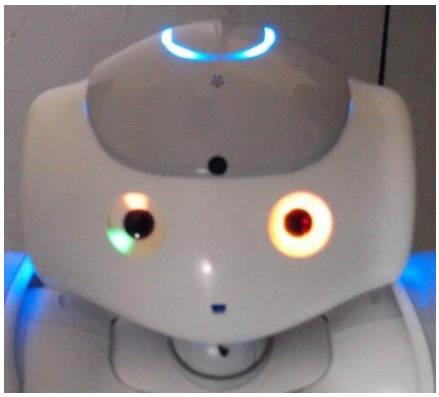


Figura 5.13: Prueba de iluminación de LEDs

A cada ojo le correspondía aportar, únicamente, la información relativa al brazo que se encontraba a su lado, es decir, el ojo derecho se iluminaría en color rojo siempre que el brazo derecho presionara una tecla cuyo sonido correspondiera a la nota “Do”. De esta forma, en el caso de interpretarse dos notas simultáneas, cada ojo se iluminaría con el color referente a la tecla pulsada por su brazo tal y como se muestra en la figura 5.14. De este modo, se consigue que la funcionalidad sea válida para cada una de las posibles acciones del robot. En el caso excepcional de la reproducción de un silencio, ambos ojos se iluminarían en color blanco para indicar que la interrupción de la melodía responde a un requerimiento de la partitura y que, por tanto, no es consecuencia de un error en el proceso.

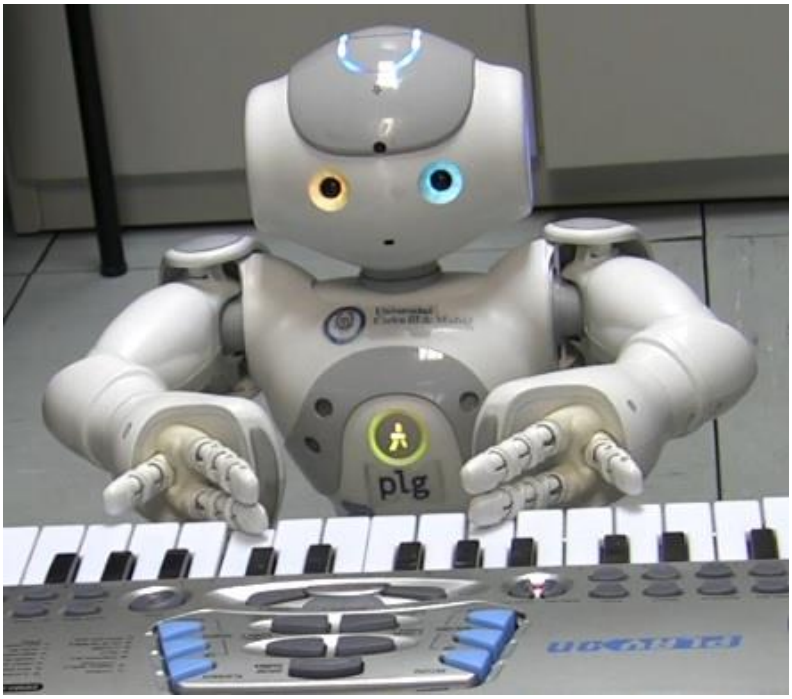


Figura 5.14: Iluminación de LEDs para notas simultáneas

Con el objetivo de facilitar el uso de esta funcionalidad, la aplicación, tras la realización de la secuencia de movimientos correspondiente al posicionamiento inicial, libera las articulaciones del cuello del robot para que la cabeza del mismo pueda ser orientada en la dirección más conveniente para el usuario.

5.3.5 - COMUNICACIÓN VERBAL Y ÓRDENES POR VOZ

Para lograr una interacción más humana con el robot, se consideró oportuno que éste tuviera las capacidades de hablar y de escuchar las instrucciones proporcionadas por el usuario. Esta modificación, además, simplifica el uso de la aplicación al no requerirse la introducción de comandos a través de un ordenador para controlar el desarrollo de la acción. En el robot NAO, las tareas de reconocimiento de voz pueden ser realizadas en la amplia variedad de idiomas mostrada en la figura 5.15 lo cual permite la múltiple traducción de la aplicación extendiendo, de este modo, las posibilidades de uso de la misma.

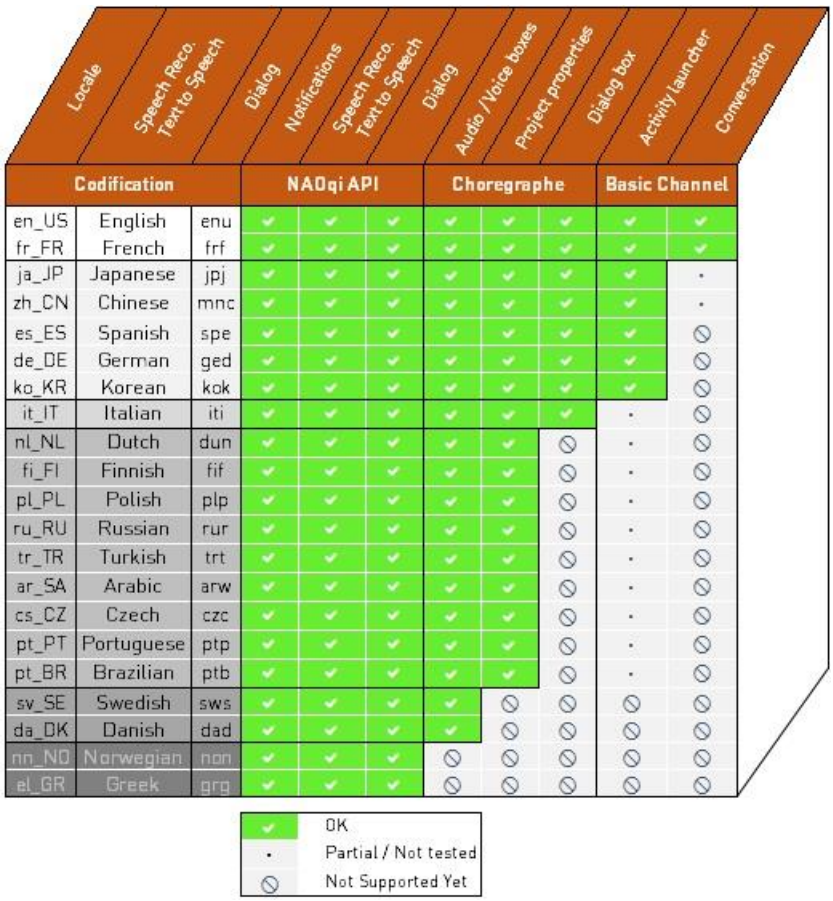


Figura 5.15: Relación de idiomas disponibles para el robot *NAO* [26]

El idioma para el óptimo reconocimiento de voz en el NAO es el francés pero, adicionalmente, se ha realizado también una versión de la aplicación en inglés por considerarse un idioma más apropiado para la presentación del proyecto al ser su uso más extendido. Aunque se hayan realizado versiones de la aplicación en dos idiomas, ambas funcionan de modo idéntico.

Alcanzado este punto del proceso, al ejecutar la aplicación final, el robot se sitúa en la postura “Crouch”, saluda y espera una respuesta. Al recibir el saludo del usuario, posiciona las manos sobre el teclado, pregunta qué melodía se desea que interprete y escucha nuevamente. Tras recibir un valor entre 1 y 5, correspondiente a la posición de la partitura en el fichero “index.txt”, pide la confirmación de dicho valor y, en caso afirmativo, interpreta la partitura seleccionada. Una vez finalizada la melodía, pregunta nuevamente qué debe interpretar. En caso de recibir una despedida en lugar del mencionado valor entre 1 y 5, el robot retorna a su posición inicial, se despide y finaliza la ejecución del *script*.

Una vez llegados a este punto, se dan por alcanzados la totalidad de los objetivos planteados al comienzo del proyecto.

5.3.6 - PRUEBAS CON CAPTACIÓN DE TEMPERATURAS

Una vez desarrollada la aplicación, se realizaron diversos test con captación de temperaturas con el objetivo de optimizar la aplicación para que pudiera utilizarse durante el máximo tiempo posible sin suponer un riesgo para los motores y sensores presentes en las diversas articulaciones del robot.

La captación de temperaturas se realiza automáticamente de forma interna en el NAO y es posible acceder de manera sencilla a estos datos conociendo la dirección IP del robot.

Dadas las características del proyecto, para que la utilización del NAO fuera óptima, las primeras articulaciones que debían presentar sobrecalentamiento se encontrarían en los brazos puesto que son las extremidades más activas durante la acción.

Tras un primer conjunto de pruebas, se encontró que las articulaciones que se calentaban de forma más rápida eran rodillas y caderas. Para mejorar el funcionamiento de la aplicación, se decidió liberar los movimientos de una rodilla y de una cadera, es decir, separarlas de sus correspondientes motores permitiendo el libre movimiento de la articulación y evitando así su calentamiento prematuro. Esta mejora se llevó a cabo tras verificar que no comprometía la posición de equilibrio del robot así como las posiciones captadas para el correcto pulsado de las teclas.

Una vez realizadas las anteriores modificaciones y tras un segundo conjunto de pruebas, se comprobó que las articulaciones con un calentamiento más rápido se encontraban en hombros y codos, es decir, ya se había alcanzado el objetivo de mejora planteado.

5.4 - CARACTERÍSTICAS DE LAS PARTITURAS EN FORMATO MUSICXML

El formato *MusicXML* permite la codificación de una gran cantidad de datos referentes a cada uno de los elementos relacionados con la partitura, como el título de la misma o el aspecto que deben tener determinadas figuras en caso de realizarse una representación gráfica. Para que al robot le sea posible interpretar la partitura, no resulta necesario disponer de la mayor parte de los datos aportados en una partitura generada a través de, por ejemplo, el software *MuseScore*. Por este motivo, para simplificar el manejo de estas partituras complejas, se han seleccionado un conjunto de etiquetas que, combinadas, aportan la información suficiente para que el NAO realice una correcta interpretación de la melodía.

Durante el proceso de lectura, la aplicación recoge únicamente datos provenientes del grupo de etiquetas elegido descartando, de este modo, todas las características que no resultan imprescindibles para la interpretación de la melodía. En la figura 5.16, se muestra el esquema del mencionado conjunto de etiquetas imprescindibles y, a continuación, se describe la función de cada una de ellas dentro de la partitura.

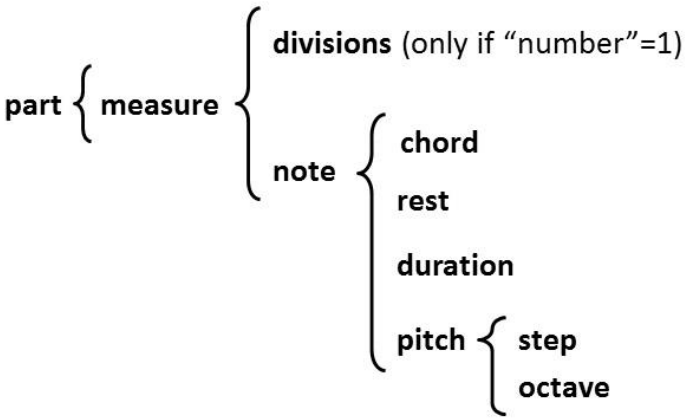


Figura 5.16: Esquema de las etiquetas utilizadas por la aplicación

<part></part>. Representa un pentagrama y, por tanto, contiene todos los elementos relacionados con la melodía. Existirán varios elementos "part" en casos en los que participen varios instrumentos por lo que, en caso de considerar cada una de ellas, la melodía se interpretaría más de una vez. Para evitar esta múltiple interpretación de un mismo tema, la aplicación únicamente considera la primera etiqueta "part" de la partitura, descartando cualquier otra del mismo tipo.

<measure></measure>. Contenida en "part", esta etiqueta representa un compás y contiene información sobre el tono y las figuras asociadas a las notas que lo componen. Adicionalmente, el primer compás de la partitura contiene datos relativos a elementos comunes como la clave o el tipo de compás utilizado. Este primer elemento "measure" aportará el valor contenido en "divisions", imprescindible para el cálculo de la duración indicada por la figura correspondiente a cada nota.

<divisions></divisions>. Esta etiqueta, incluida únicamente en el primer "measure", contiene un valor numérico que es indicativo del número de divisiones realizadas sobre el valor temporal de una figura negra, es decir, la fracción temporal correspondiente a cada nota deberá aparecer multiplicada por este valor. Mediante el uso de esta etiqueta, se consigue que el valor asociado a la duración de cada nota sea siempre un número entero.

<note></note>. Esta etiqueta aparece siempre dentro de otra de tipo “measure” y representa una nota siendo, por tanto, el elemento donde se incluyen los datos que permiten obtener el tono y la figura asociados a la misma.

<duration></duration>. Siempre en el interior de “note”, contiene un valor numérico que indica la figura musical correspondiente. Este valor está directamente relacionado, tal y como se ha mencionado anteriormente, con el contenido de “divisions”. La aplicación toma, como valor de tiempo básico, la duración temporal estimada para una figura negra y lo multiplica por el valor obtenido de (“duration”/”divisions”) obteniendo así el tiempo durante el cual debe emitirse el sonido.

<pitch></pitch>. Contenida en “note”, aporta los datos relativos al tono, es decir, indica el nombre de la nota dentro de la escala y su octava correspondiente. El conjunto de la información incluida en esta etiqueta permite la localización de la tecla que debe ser pulsada.

<step></step>. Se encuentra en el interior de “pitch” y contiene una letra que puede ser “A”, “B”, “C”, “D”, “E”, “F” o “G” siendo estos valores correspondientes, respectivamente, a las notas “La”, “Si”, “Do”, “Re”, “Mi”, “Fa” y “Sol”.

<octave></octave>. Se incluye dentro de “pitch” y aporta un valor numérico que complementa al dato obtenido del interior de la etiqueta “step” al indicar la octava correspondiente a la nota obtenida, es decir, al concretar en qué punto exacto del pentagrama se encuentra. Esta información resulta imprescindible debido a que el teclado contiene múltiples escalas y, por tanto, varias notas con el mismo nombre.

<rest/>. Incluida en “note”, no tiene contenido pero, con su presencia, indica que la nota a interpretar es un silencio. Por este motivo, resulta evidente que las etiquetas “rest” y “pitch” no pueden aparecer juntas dentro del mismo elemento de tipo “note” ya que las informaciones que aportan son incompatibles.

<chord/>. Contenida en “note”. Aunque no tiene contenido, complementa la información aportada por una etiqueta “pitch” al indicar que, el sonido indicado, debe comenzar al mismo tiempo que la última nota que no disponía de una etiqueta “chord”.

En la figura 5.17 se muestra un ejemplo de código *MusicXML* en el cual se utilizan únicamente las etiquetas legibles por la aplicación. En este caso, se solicitaría la interpretación simultánea de las notas “Do” y “Sol” durante el tiempo equivalente a una figura negra seguidas de un silencio de idéntica duración.

```
<part id="P1">
        <measure number="1">
                <divisions>1</divisions>
                <note>
                        <pitch>
                                <step>C</step>
                                <octave>4</octave>
                        </pitch>
                        <duration>1</duration>
                </note>
                <note>
                        <chord/>
                        <pitch>
                                <step>G</step>
                                <octave>4</octave>
                        </pitch>
                        <duration>1</duration>
                </note>
                <note>
                        <rest/>
                        <duration>1</duration>
                </note>
        </measure>
</part>
```

Figura 5.17: Ejemplo de código *MusicXML* simplificado

6 - MANUAL DE USUARIO

El presente manual sirve como referencia para el correcto uso de las funcionalidades desarrolladas durante el proyecto.

La aplicación está compuesta por la carpeta “Partituras” y por, al menos, un archivo cuyo nombre viene determinado por tres letras, identificativo del idioma utilizado, seguidas de la palabra “Piano”, esto es, “EngPiano.py” para la versión inglesa o “FraPiano.py” para la versión francesa.

6.1 - LA CARPETA “PARTITURAS”

Este directorio es el contenedor de los archivos de partitura en formato *MusicXML* que pueden ser creados de forma sencilla mediante el uso del software libre *MuseScore*. Debe tenerse en cuenta que únicamente será válido el primer pentagrama de la partitura y que, en caso de utilizarse alteraciones, éstas no serán reconocidas, es decir, la aplicación considerará que “La”, “La sostenido” y “La bemol” son notas idénticas puesto que no dispone de las coordenadas necesarias para alcanzar las teclas negras asociadas a las notas alteradas. Los archivos generados deben tener una denominación del tipo “Nombre.xml”.

Dado que el diseño de la aplicación únicamente permite la selección de la melodía de entre una lista de 5 opciones, es necesario indicar, a través de “Index.txt”, cuáles de los archivos de partitura deben estar incluidos en dicha lista para ser reconocidos por el robot. En “Index.txt” deben escribirse los nombres, sin incluir el tipo de extensión “.xml”, de los 5 archivos seleccionados debiendo ocupar, cada uno de ellos, una línea completa del documento. Además, para indicar el final de la lista en caso de contener la misma menos de 5 elementos, debe incluirse una línea extra al final de “Index.txt” que contenga únicamente un corchete de cierre “}“ (es recomendable no retirar esta línea adicional aunque se utilicen 5 opciones).

En la figura 6.1 se muestra un ejemplo de los posibles contenidos de la carpeta “Partituras” y del documento “Index.txt”.

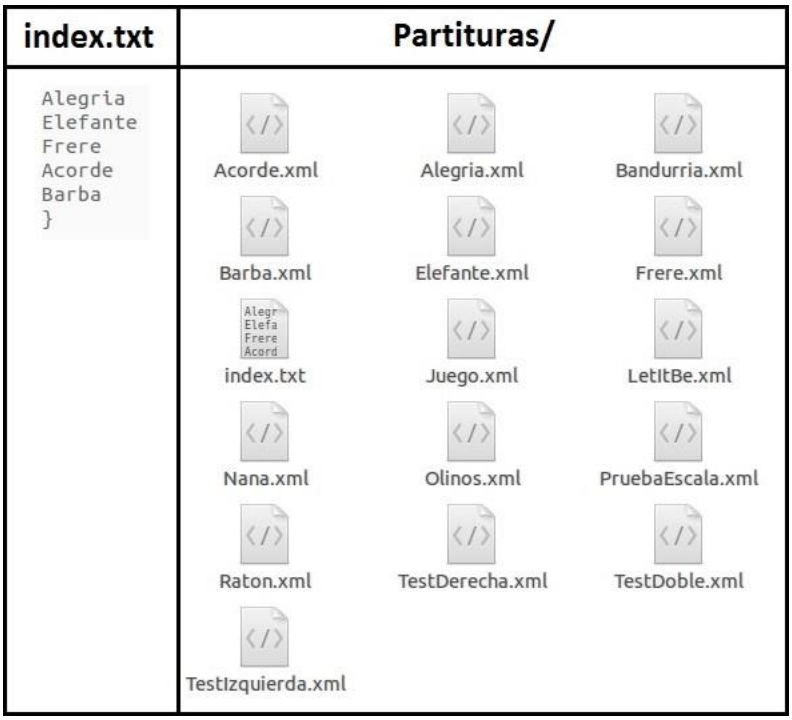


Figura 6.1: Ejemplo de un archivo “index.txt” y la carpeta “Partituras” que lo contiene

6.2 - USO DE LA APLICACIÓN

Al ejecutar el fichero que complementa a la carpeta “Partituras” el robot NAO se situará en la postura que se muestra en la figura 6.2 y realizará una presentación de los colores que serán utilizados, en los LEDs de sus ojos, para proporcionar información relativa a la nota interpretada en cada momento. Finalizada esta presentación, saludará y esperará una respuesta.

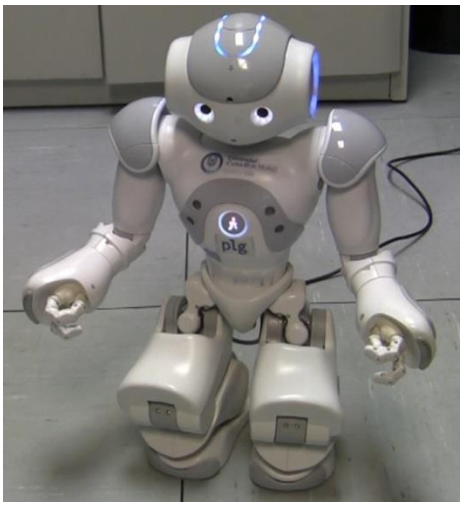


Figura 6.2: Postura inicial

Tras escuchar “hola”, el robot pasará a una postura con los brazos levantados a la altura de sus hombros. En este punto, el teclado debe ser situado de tal forma que cada una de las manos esté directamente situada sobre la tecla central de la mitad de instrumento que le corresponde, es decir, cada uno de los dos dedos más próximos al teclado, los encargados de pulsar las teclas, deberán quedar situados sobre la sexta tecla contada desde el extremo que se encuentre más próximo, tal y como se muestra en la figura 6.3.

Figura 6.3: Posicionamiento correcto del teclado

Una vez haya realizado este segundo posicionamiento, preguntará qué se quiere escuchar y esperará nuevamente la respuesta del usuario. La forma de seleccionar la melodía a interpretar es decir un número entre “uno” y “cinco” correspondiente a la posición que ocupa esta partitura dentro del fichero “Index.txt”. Para verificar que el número ha sido escuchado correctamente, el robot solicitará una confirmación repitiendo la selección realizada. Si la respuesta a esta confirmación es “no”, retornará a la selección de partitura mientras que, si dicha respuesta es “si”, realizará la interpretación solicitada para, posteriormente, solicitar de nuevo una melodía a interpretar.

Durante toda la interpretación, la cabeza del robot puede orientarse libremente para visualizar de manera más sencilla la información proporcionada por sus ojos, cada uno de los cuales se iluminará indicando la nota que está interpretando su brazo correspondiente. La nota correspondiente a cada color es idéntica a la expuesta por el propio robot al inicio de la ejecución de la aplicación, inmediatamente después del primer posicionamiento.

En caso de no querer proseguir con la interpretación de más melodías, en lugar de indicar un número en la selección de partitura, debe decirse “adiós” en el idioma que corresponda a la aplicación utilizada. Tras escuchar la despedida, el robot retirará los brazos de su posición sobre el teclado, se despedirá y finalizará la ejecución de la aplicación.

7 - MANUAL DE REFERENCIA

Con el objetivo de simplificar la introducción de la aplicación en el robot para su automática ejecución, ésta se ha desarrollado en un único *script* cuya estructura se describe en el presente manual de referencia.

Todo el código ha sido desarrollado en lenguaje *Python* y está compuesto por métodos de una única clase global y una subclase necesaria para la gestión de los eventos generados durante el reconocimiento de voz.

Tras la importación de los módulos necesarios, se encuentran los datos identificativos del robot y la creación de sus *Proxys*.

Antes de la definición de los métodos a utilizar, se crean las variables más relevantes del programa, esto es, las variables que indican la posición relativa de cada una de las teclas así como la de las combinaciones entre ellas y las variables encargadas de regular el flujo de la aplicación. Antes de la creación de las variables de flujo, se introducen el idioma y las palabras a identificar por el módulo de reconocimiento de voz.

Las primeras funciones definidas son las encargadas de modificar las variables de flujo. Estos métodos serán llamados únicamente por la clase "ALWords" que se crea posteriormente, junto con el "broker" necesario para la correcta gestión de los eventos. Debe tenerse en cuenta que, en *Python*, para posibilitar la modificación de las variables declaradas fuera de una función, es necesario incluir una nueva declaración de variable de tipo "global" accediendo, de este modo, a las variables declaradas anteriormente.

Los métodos "sentadilla" y "posicionamiento" sitúan al robot en la postura escogida para la realización de la acción y sus variables de posición se han creado dentro se las propias funciones para facilitar la edición de las mismas. De la misma forma, el método "retornoReposo" se encarga de volver a situar al robot en una postura que requiera un menor esfuerzo de los hombros del robot momentos antes de la finalización de la aplicación.

La lectura, conversión e interpretación de las partituras corre a cargo del método "interpretar" que, dada su complejidad, ha sido dividido en varias funciones para facilitar sus *test* de funcionamiento.

Dado que las partituras se encuentran fuera del *script*, en otro directorio, se almacenan en una variable los nombres de aquellas que son susceptibles de ser leídas por la aplicación. Para esto, el método "leerIndex", llamado inmediatamente tras la creación de la variable mencionada, busca el fichero "Index.txt" y extrae la información presente en el mismo.

Por último, el método “run” regula todo el flujo de la aplicación puesto que es el encargado de controlar la suscripción a los eventos de reconocimiento de voz y la realización de las acciones requeridas a través de dichos eventos. La llamada a esta función se realiza desde la clase principal al iniciarse la aplicación.

8 – GESTIÓN DEL TRABAJO

A lo largo de este capítulo se analizan los recursos y el tiempo requeridos para llevar a cabo el completo desarrollo de la aplicación.

En primer lugar, se ha analizado la planificación del proyecto con el objetivo de conocer cuáles han sido las tareas realizadas y cuáles son las relaciones de precedencia entre ellas para, posteriormente, calcular la cuantía del coste económico total del proyecto.

8.1 - PLANIFICACIÓN

El proyecto ha sido subdivido en cuatro tareas de las cuales, las dos primeras, están divididas, a su vez, en otros conjuntos de subtareas. A continuación, se exponen cuáles son estas cuatro tareas principales y su cometido:

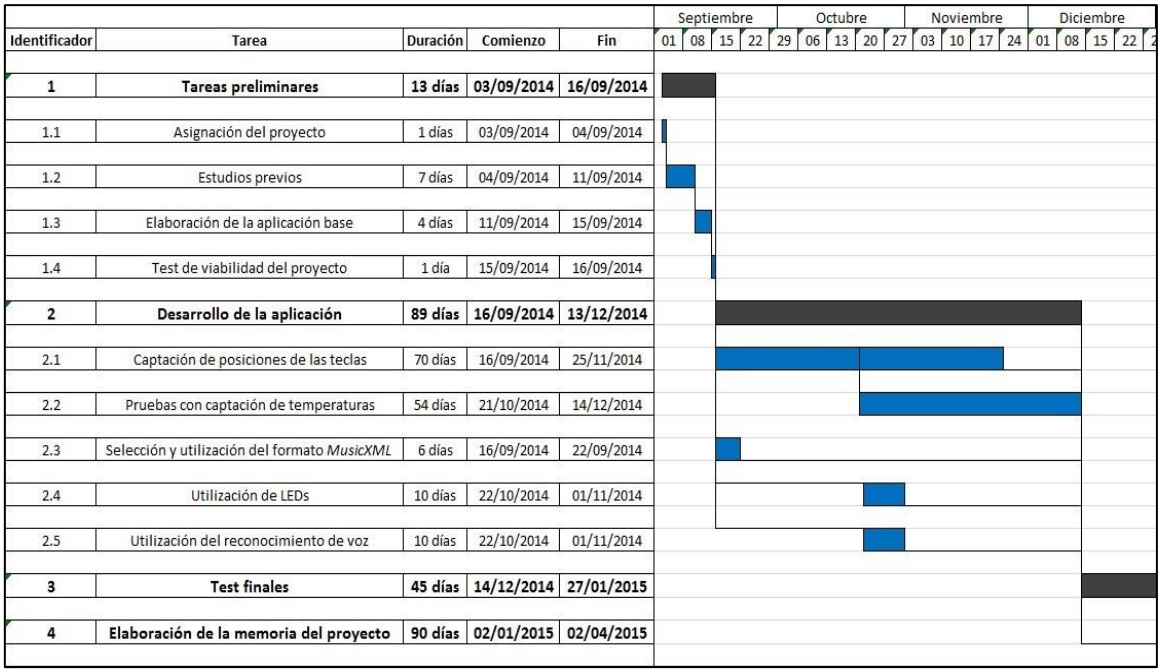
- **Tareas preliminares.** Comprenden el estudio de los objetivos a alcanzar y las comprobaciones orientadas a verificar que el desarrollo del proyecto es viable, es decir, que las características del hardware del robot NAO le permiten realizar correctamente la acción deseada.

- **Desarrollo de la aplicación.** Tras haber verificado que el proyecto es viable, pueden comenzar las tareas correspondientes al desarrollo e inclusión en la aplicación principal de cada una de las funcionalidades propuestas para alcanzar los objetivos propuestos durante la tarea anterior.

- **Test finales.** Una vez completado el desarrollo de la aplicación, ésta debe ser probada en repetidas ocasiones con objeto de garantizar que su funcionamiento es el adecuado.

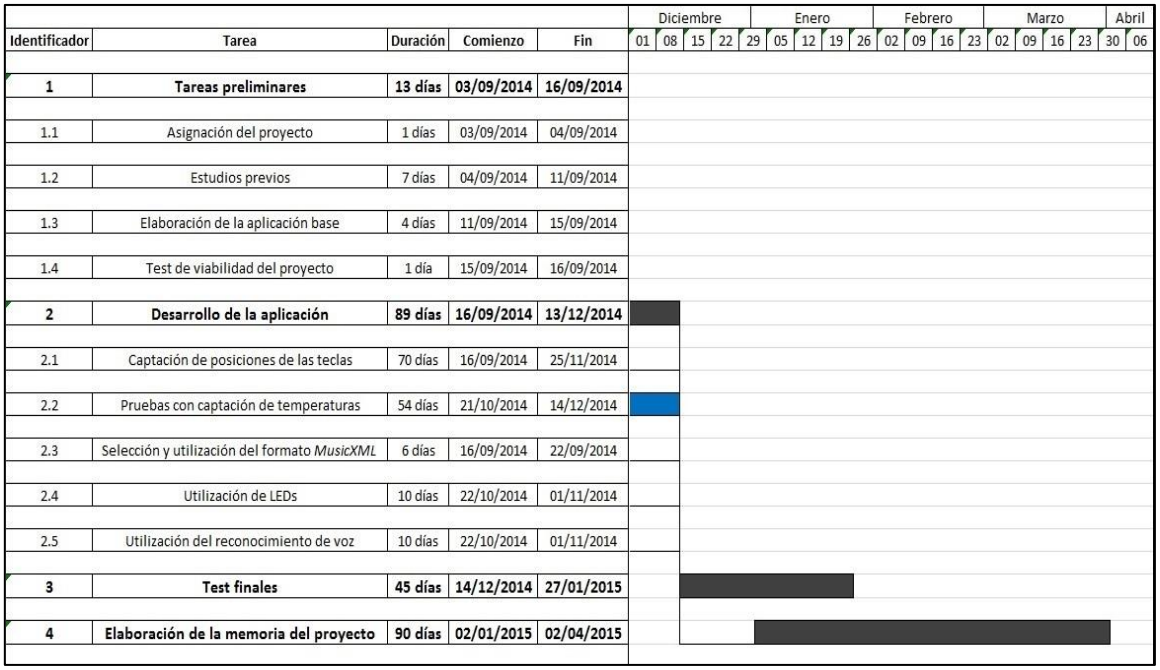
- **Elaboración de la memoria del proyecto.** A lo largo de esta última tarea, se desarrolla la presente memoria de proyecto y, aunque puede ser elaborada una vez finalizado el desarrollo de la aplicación, se ha considerado más conveniente que su realización comience tras haberse realizado, al menos, un test completo de funcionamiento.

A continuación, se muestra el diagrama de Gantt correspondiente a esta planificación donde las tareas principales aparecen marcadas en gris y sus subtareas en azul. Para mejorar la visibilidad de este diagrama se han representado por separado los meses Septiembre-Diciembre en la figura 8.1 y los meses Diciembre-Abril en la figura 8.2.



Identificador	Tarea	Duración	Comienzo	Fin
1	**Tareas preliminares**	**13 días**	**03/09/2014**	**16/09/2014**
1.1	Asignación del proyecto	1 días	03/09/2014	04/09/2014
1.2	Estudios previos	7 días	04/09/2014	11/09/2014
1.3	Elaboración de la aplicación base	4 días	11/09/2014	15/09/2014
1.4	Test de viabilidad del proyecto	1 día	15/09/2014	16/09/2014
2	**Desarrollo de la aplicación**	**89 días**	**16/09/2014**	**13/12/2014**
2.1	Captación de posiciones de las teclas	70 días	16/09/2014	25/11/2014
2.2	Pruebas con captación de temperaturas	54 días	21/10/2014	14/12/2014
2.3	Selección y utilización del formato *MusicXML*	6 días	16/09/2014	22/09/2014
2.4	Utilización de LEDs	10 días	22/10/2014	01/11/2014
2.5	Utilización del reconocimiento de voz	10 días	22/10/2014	01/11/2014
3	**Test finales**	**45 días**	**14/12/2014**	**27/01/2015**
4	**Elaboración de la memoria del proyecto**	**90 días**	**02/01/2015**	**02/04/2015**

Figura 8.1: Diagrama de Gantt (Septiembre-Diciembre)



Identificador	Tarea	Duración	Comienzo	Fin
1	**Tareas preliminares**	**13 días**	**03/09/2014**	**16/09/2014**
1.1	Asignación del proyecto	1 días	03/09/2014	04/09/2014
1.2	Estudios previos	7 días	04/09/2014	11/09/2014
1.3	Elaboración de la aplicación base	4 días	11/09/2014	15/09/2014
1.4	Test de viabilidad del proyecto	1 día	15/09/2014	16/09/2014
2	**Desarrollo de la aplicación**	**89 días**	**16/09/2014**	**13/12/2014**
2.1	Captación de posiciones de las teclas	70 días	16/09/2014	25/11/2014
2.2	Pruebas con captación de temperaturas	54 días	21/10/2014	14/12/2014
2.3	Selección y utilización del formato *MusicXML*	6 días	16/09/2014	22/09/2014
2.4	Utilización de LEDs	10 días	22/10/2014	01/11/2014
2.5	Utilización del reconocimiento de voz	10 días	22/10/2014	01/11/2014
3	**Test finales**	**45 días**	**14/12/2014**	**27/01/2015**
4	**Elaboración de la memoria del proyecto**	**90 días**	**02/01/2015**	**02/04/2015**

Figura 8.2: Diagrama de Gantt (Diciembre-Abril)

El proyecto se ha desarrollado durante 237 días distribuidos a lo largo de 8 meses.

8.2 - PRESUPUESTO

Para realizar el análisis del presupuesto del proyecto se han calculado, de forma independiente, los costes directos relativos al personal y al material necesario. Adicionalmente, para obtener el coste total, se ha fijado que los costes indirectos, correspondientes a diversas facturas como luz o conexión a internet, tienen un valor del 20% del total de los costes directos.

8.2.1 - COSTES DE PERSONAL

Para el cálculo de estos costes se ha considerado un coste por trabajador de 25€/hora, habiendo participado un único operario en el desarrollo del proyecto.

Aunque la distribución de las horas de trabajo ha sido bastante desigual a lo largo de cada una de las tareas, se ha estimado que el tiempo de dedicación del empleado, sin incluir transportes o descansos, ha sido de 400 horas.

En la figura 8.3 se muestra la tabla de cálculo de los costes de personal.

	Concepto	Cantidad	Coste(€)/hora	Horas de dedicación	Coste(€)
	Empleados	1,00	25,00	400,00	10.000,00
Total		**1,00**	**25,00**	**400,00**	**10.000,00**

Figura 8.3: Costes de personal

8.2.2 - COSTES DE MATERIAL

Al calcular el coste de material debe tenerse en cuenta la amortización de cada uno de los elementos utilizados. Debido a esto, para la obtención del coste imputable al equipo utilizado, se ha empleado la siguiente fórmula:

$$Coste\ imputable = \frac{Tiempo\ de\ utilización}{Tiempo\ de\ vida\ estimado} * Coste\ unitario$$

A continuación, se exponen los componentes del equipo requerido y sus respectivos tiempos de utilización y vida que, posteriormente, serán utilizados para el cálculo de su coste:

- **Ordenador portátil.** Se utiliza a lo largo de los ocho meses de proyecto y su tiempo de vida es de, aproximadamente, tres años.

- **Robot NAO.** A diferencia del ordenador, el robot no es necesario durante los meses de Febrero y Marzo puesto que la única tarea en curso es la elaboración de la presente memoria de proyecto por lo que su tiempo de utilización es de seis meses. Se ha considerado que el tiempo de vida del robot NAO es de tres años.

- **Teclado electrónico.** Dado que ha sido adquirido con el propósito único de ser utilizado en este proyecto, se ha decidido considerar que el tiempo de vida del teclado es idéntico a su tiempo de utilización, es decir, su coste imputable es su coste unitario.

- **Mesa soporte.** Ha sido diseñada y fabricada para su utilización en este proyecto por lo que, al igual que en el caso del teclado, se ha establecido que su coste imputable es equivalente a su coste unitario. Adicionalmente, dado que el tiempo de utilización de las herramientas requeridas durante el proceso de fabricación es despreciable en relación al tiempo de vida de las mismas, se ha establecido que el coste unitario del soporte es, aproximadamente, el coste de los materiales que lo componen.

En la figura 8.4 se muestra la tabla de cálculo de los costes de material.

	Concepto	Coste(€)	Coste imputable(€)
	Ordenador portátil (4Gb RAM; 1,6 GHz)	500,00	111,11
	Robot NAO + licencia *Choregraphe*	5.675,00	945,83
	Teclado electrónico	30,00	30,00
	Materiales de la mesa soporte	15,00	15,00
Total		**6.220,00**	**1.101,94**

Figura 8.4: Costes de material

8.2.3 - COSTE TOTAL

Para obtener el coste total del proyecto deben añadirse, a los costes directos ya calculados, los costes indirectos obtenidos de la siguiente ecuación:

$$Costes\ indirectos = 0.2 * (Costes\ de\ personal + Costes\ de\ material)$$

En la figura 8.5 se muestra la tabla de cálculo del coste total del proyecto.

	Concepto	Coste imputable(€)
	Personal	10.000,00
	Material	1.101,94
Subtotal		**11.101,94**
	Costes Indirectos	2.220,39
Total		**13.322,33**

Figura 8.5: Coste total

El presupuesto necesario para realizar el proyecto es de TRECE MIL TRESCIENTOS VEINTIDÓS EUROS CON TREINTA Y TRES CÉNTIMOS (13.322,33 €).

9 - RESULTADOS

Una vez finalizado el proyecto, se puede afirmar que se han alcanzado la totalidad de los objetivos propuestos al comienzo del mismo.

Durante la elaboración de la aplicación, se ha verificado el funcionamiento de cada una de las funciones creadas, antes de su inclusión en el código principal, con el objetivo de minimizar la posibilidad de encontrar errores en la ejecución global, donde su detección y corrección resulta más compleja.

Para garantizar el correcto funcionamiento de la aplicación, se ha decidido establecer una serie de test de funcionamiento que la aplicación debe superar con éxito a lo largo de su desarrollo. Estos test básicos son:

- Test de doble escala
- Test de notas dobles
- Test final de la aplicación

Tras la finalización del proyecto, la aplicación había superado con éxito la totalidad de los test básicos de funcionamiento propuestos por lo que, además de asegurar que se han alcanzado los objetivos planteados, es posible afirmar que los requisitos establecidos se cumplen de forma estable a lo largo de múltiples ejecuciones de la aplicación.

9.1 – TEST DE DOBLE ESCALA

Este test fue realizado tras la captación de las posiciones que permiten al robot alcanzar las teclas y su objetivo es garantizar que dichas posiciones son válidas para su uso en múltiples ejecuciones del código.

Consiste en la interpretación de una escala completa ascendente y otra descendente, es decir, el pulsado consecutivo de cada una de las 22 teclas de izquierda a derecha y viceversa. El motivo de este procedimiento es que el error más significativo detectado en los movimientos de los brazos se produce en pequeños desplazamientos, esto es, el pulsado de teclas sucesivas es la acción más susceptible de generar un fallo en la interpretación.

Para garantizar su validez, este test debe ser realizado, al menos, dos veces consecutivas durante una única ejecución de la aplicación y una vez más tras realizarse un reinicio del robot y haberse modificado la ubicación del mismo. La primera prueba verifica que las posiciones empleadas son válidas para el robot y teclado utilizados, la

segunda certifica que, durante la interpretación, los movimientos realizados no son lo suficientemente bruscos para comprometer la validez de las posiciones empleadas y, por último, la tercera garantiza que el sistema es robusto frente a pequeñas variaciones del terreno como, por ejemplo, un cambio del tipo de suelo o la presencia de imperfecciones considerables en el mismo.

9.2 – TEST DE NOTAS DOBLES

Este test fue realizado después del test de doble escala y su objetivo es verificar que, al interpretar notas simultáneas, ambos brazos actúan de forma sincronizada sin comprometer la validez de las posiciones empleadas.

Consiste en la interpretación de una sucesión de pares de notas. A diferencia del caso del test anterior, no es necesario que se cubran todas las posibilidades puesto que, dado que ya se conoce que las posiciones individuales son válidas, un buen funcionamiento de una serie de pares aleatorios es suficiente indicativo del comportamiento del sistema.

Al igual que el test de doble escala, este test debe realizarse, al menos, dos veces consecutivas durante una única ejecución de la aplicación y una vez más tras realizarse un reinicio del robot y haberse modificado la ubicación del mismo. La primera prueba garantiza que las posiciones individuales utilizadas para crear las dobles son válidas sin necesidad de modificación adicional alguna, la segunda verifica que, aunque ambos brazos se desplacen de manera conjunta, los movimientos aún no son lo suficiente buscos para producir una variación de la postura que comprometa la validez de las posiciones empleadas y, por último, la tercera certifica nuevamente la robustez del sistema frente a leves variaciones en las características del terreno.

9.3 – TEST FINAL DE LA APLICACIÓN

Este test fue realizado tras la finalización del proyecto y su objetivo es garantizar el correcto funcionamiento del conjunto de funciones que pueden ser ejecutadas por la aplicación.

Consiste en la ejecución de la aplicación tal y como la realizaría un posible usuario de la misma. Sin embargo, dado que se pretende verificar el buen funcionamiento del código, deben utilizarse todas las palabras susceptibles de ser captadas por el reconocimiento de voz así como un conjunto de partituras que requieran la interpretación de la mayor variedad de notas y figuras posible.

Este test debe ser realizado, al menos, dos veces para cada uno de los idiomas utilizados y debe poder garantizar que:

- **El reconocimiento de voz funciona correctamente**. Las palabras empleadas para la interacción con el robot deben diferenciarse correctamente y ser reconocidas sin necesidad de ser repetidas de manera excesiva.

- **Los LEDs se sincronizan de forma correcta**. Los LEDs deben proporcionar la información relativa a la nota que se está interpretando únicamente durante el proceso de pulsado de la tecla.

- **La melodía interpretada es reconocible**. El tiempo correspondiente a cada figura musical debe ser respetado de tal forma que se cree una melodía reconocible sin importar la velocidad de los movimientos del robot *NAO*.

Para exponer con mayor claridad los resultados obtenidos se han elaborado dos vídeos a cuya visualización puede accederse a través de los siguientes enlaces:

- El primero de estos vídeos muestra la realización de un test de doble escala completo seguido de un breve test de notas dobles.

  https://www.youtube.com/watch?v=B3KwizEDdLQ

- En el segundo vídeo puede observarse una demostración del funcionamiento de la aplicación en inglés.

  https://www.youtube.com/watch?v=-b0HGQf6BiA

10 - CONCLUSIONES

A la vista de los resultados obtenidos, es posible concluir que la aplicación desarrollada a lo largo de este proyecto cumple de manera efectiva su propósito de generar entretenimiento mediante el uso de un robot *NAO*.

Aunque globalmente la aplicación posee las características deseadas, se ha observado que la velocidad de movimiento del robot *NAO* utilizado durante este proyecto no es lo suficientemente elevada para interpretar, con la calidad deseable, partituras con figuras musicales de corta duración.

A pesar de esta carencia, el robot puede cumplir una función didáctica gracias a la información proporcionada a través de sus LEDs y a la sencillez de los movimientos que realiza. Para un potencial alumno, la observación de las teclas pulsadas por el *NAO*, junto con el conocimiento de las notas interpretadas, puede resultar de gran interés y ayuda para obtener los conocimientos básicos necesarios para la utilización de un teclado. Adicionalmente, puesto que la creación y edición de partituras puede ser realizada con facilidad, el alumno dispone de la posibilidad de aprender cómo interpretar sus melodías favoritas relacionando, al mismo tiempo, los elementos presentes en el pentagrama con su significado sobre el teclado.

Cabe destacar que la versión del cuerpo del robot *NAO* utilizada es la V3.2 y, actualmente, existen versiones más avanzadas del mismo por lo que es de esperar que, en caso de utilizarse un robot algo más evolucionado, su velocidad de movimientos se vea incrementada y, por tanto, mejore la calidad de las melodías interpretadas.

11 - LÍNEAS FUTURAS

A continuación, se exponen algunas de las posibles líneas futuras para la mejora de esta aplicación.

- **Incorporación de un comando de parada urgente.** Durante la interpretación de una melodía, el usuario no dispone de ningún comando de voz o botón válido, es decir, no es posible detener la interpretación de una melodía que ya ha comenzado. Dado que las partituras pueden incluirse sin importar su longitud, la posibilidad de poner fin a la interpretación de forma manual puede resultar necesaria en algunos casos.

- **Captación automática de temperaturas.** Tras un uso prolongado, el robot podría experimentar un sobrecalentamiento en algunas de sus articulaciones. Si esto ocurre, el *NAO* informa verbalmente de su situación permitiendo que el usuario pueda retirar a tiempo el teclado antes del comienzo de la rutina automática de apagado. Para que no se requiera la intervención de ningún agente externo, el robot debería retornar a una posición segura lejos del teclado antes de que se produzca un aumento de temperatura excesivo. Para alcanzar este objetivo, se sugiere la realización de una captación interna de temperaturas en las articulaciones críticas de manera periódica para que, al detectar el mencionado calentamiento, se detenga la interpretación en curso y se ponga fin a la ejecución de la aplicación.

- **Utilización de un feedback auditivo.** Para garantizar que la acción se está desarrollando correctamente, se propone que el robot escuche y analice la melodía que está interpretando para verificar la validez de los tonos emitidos.

- **Utilización del módulo de visión.** La aplicación, en su estado actual, requiere la intervención directa del usuario para situar el teclado en su posición pero, mediante la utilización de las cámaras incorporadas en el *NAO*, éste podría llegar a posicionarse de forma autónoma frente al teclado.

- **Adición de un modo "improvisación".** Para que la aplicación resulte aún más atractiva, se sugiere la creación de un algoritmo que permita al robot crear melodías originales. Con este nuevo modo de utilización, el usuario podría solicitar una improvisación por parte del *NAO* y obtener un fichero de partitura con la nueva melodía creada para que ésta pueda ser modificada y reinterpretada.

- **Utilización de varios robots *NAO*.** Mediante la creación de nuevas aplicaciones, es posible la creación de un pequeño grupo de robots músicos que creen una melodía más elaborada mediante el uso de diversos instrumentos musicales. Para alcanzar este objetivo se propone que, la aplicación actual, tras haber leído y convertido el formato para cada uno de los instrumentos, envíe las notas de manera sincronizada a cada uno de los robots para éstos realicen una interpretación coordinada.

12 - BIBLIOGRAFÍA

Para mostrar la bibliografía utilizada durante la realización de este proyecto con mayor claridad, se ha decidido realizar una división entre las fuentes de las cuales se ha extraído información y aquellas de las que únicamente se han obtenido figuras.

12.1 - FUENTES DE INFORMACIÓN

[1] http://www.rae.es/

Visitado 14/02/2015

[2] http://www.grupoisis.uma.es/microbot/public/robots.pdf

Visitado 05/03/2015

[3] http://www.etsisi.upm.es/museo_virtual/origenes/jmjacquard

Visitado 14/02/2015

[4] http://world.honda.com/ASIMO/

Visitado 05/03/2015

[5] https://www.aldebaran.com/en/robotics-company/history

Visitado 05/03/2015

[6] https://www.aldebaran.com/en/humanoid-robot/nao-robot

Visitado 14/02/2015

[7] http://doc.aldebaran.com/2-1/index.html

Visitado 16/02/2015

[8] http://ocw.uc3m.es/ingenieria-telematica/aplicaciones-moviles/material-de-clase-2/pys60

Visitado 19/02/2015

[9] https://docs.python.org/2/tutorial/

Visitado 19/02/2015

[10] http://www.musicxml.com/

Visitado 21/02/2015

[11] http://www.w3schools.com/xml/xml_whatis.asp

Visitado 21/02/2015

[12] http://www.musicxml.com/tutorial/

Visitado 21/02/2015

[13] http://musescore.org/en/musescore-tour-1

Visitado 22/02/2015

[14] http://hrl.informatik.uni-freiburg.de/papers/maierd14humanoids.pdf

Visitado 05/03/2015

[15] http://www.toyota.co.jp/en/news/07/1206_2.html

Visitado 05/03/2015

12.2 - FIGURAS

[16] http://fr.wikipedia.org/wiki/M%C3%A9tier_Jacquard

Visitado 06/03/2015

[17] https://computacionaplicadaf.files.wordpress.com/2013/03/robot-unimate.jpg

Visitado 06/03/2015

[18] http://infolab.stanford.edu/pub/voy/museum/pictures/display/1-Robot.htm

Visitado 06/03/2015

[19] http://en.wikipedia.org/wiki/Programmable_Universal_Machine_for_Assembly

Visitado 20/03/2015

[20] http://doc.aldebaran.com/2-1/family/nao_h25/dimensions_h25_v32.html#h25-dimensions-v32

Visitado 22/03/2015

[21] http://doc.aldebaran.com/2-1/family/nao_dcm/actuator_sensor_names.html

Visitado 20/03/2015

[22] http://doc.aldebaran.com/2-1/family/nao_h25/index_h25.html

Visitado 20/03/2015

[23] *Choregraphe*

[24] http://www.musicxml.com/UserManuals/MusicXML/MusicXML.htm#Tutorial.htm

Visitado 22/03/2015

[25] MuseScore

[26] http://doc.aldebaran.com/2-1/family/robots/languages.html#language-codes

Visitado 16/02/2015